Technical Manual **30/04/2021**

Institute of Technology Carlow

Institiúid Teicneolaíochta Cheatharlach

Application Green Quake

Green Quake

Student Name: Peter Lucan
Student Number: C00228946
Supervisor: Chris Meudec

Github Link:
https://github.com/PeterX12/Application-Green-Quake.git
APK Github Link:
https://github.com/PeterX12/Application-Green-Quake/blob/master/ApplicationGreenQuake.apk
APK Google Drive Link:
https://drive.google.com/file/d/1MlPChFA_Z_XOzcy3yi_Q1Gt6JyNJcC1e/view?usp=sharing

Abstract

The aim of this manual is to document the technical aspects of the Green Quake Project. This document contains the documentation of the Doxygen documentation, the installation procedure, the application screens, the database structure and the code written and designed for this project. The amount of code required was found to be quite large.

Table of Contents

1 Introduction **3**

2 Documentation **3**

3 Installation **3**

4 Application Screenshots **4**

4.1 Login Screen 4
4.2 Login Screen With Error Messages 5
4.3 Sign Up Screen 5
4.4 Sign Up Screen With Error Messages 6
4.5 Forgot Password Screen 6
4.6 Forgot Password Screen With Error Messages 7
4.7 Main Menu Screen 7
4.8 Categories Screen 8
4.9 Food and Drink Subcategories Screen 8
4.10 H20 Action Screen 9
4.11 Refill Stations Screen 9
4.12 Leaderboard Screen 10
4.13 Leaderboard Screen When The Filter Button Is Tapped 10
4.14 Leaderboard Pop Up Screen When A Profile Is Tapped On The Leaderboard 11
4.15 Filtered Leaderboard Screen 11
4.16 Profile Screen 12
4.17 Popup Screen After The Profile Image Is Tapped 12
4.18 The Trophies Screen 13
4.19 The Achievements Screen 13
4.20 The BadgesScreen 14
4.21 The Badge Popup Screen 14
4.22 The Live Avatar Mosaics on Lvl 56 15
4.23 The Live Avatar Mosaics on Lvl 57 15
4.24 The Live Avatar Mosaics on Lvl 58 16
4.25 The Live Avatar Mosaics on Lvl 59 16
4.26 The Live Avatar Mosaics on Lvl 60 17

5 Database **17**

5.1 Firebase Database 18
5.2 Firebase Storage 21
5.3 Firebase Authentication 22

6 Code Listings **22**

1 Introduction

This documentation discusses all the main technical points of the project. The Doxygen Documentation and installation steps are discussed in detail at the beginning of the Manual. This is immediately followed by the Screens of the application and the Structure of the databases that was designed to be sued for this project. List but not least is the code listings which are all the code files that were written during the development of this project.

2 Documentation

The entire project code was documented thoroughly using Doxygen and a Doxyfile config file was created and saved for this documentation. The Documentation can be found on my github. Here is the link to the Documentation:

- https://github.com/PeterX12/Application-Green-Quake/tree/master/Doxygen%20Documentation

The documentation documents the code in all the available formats. It comes in html, docbook, latex, man, rtf and xml. It is stored in the root folder of the project along with the other important files.

The doxyfile that comes with the documentian can be run using doxywizard to reproduce the documentation once if you have the entire repository provided downloaded on your local machine.

3 Installation

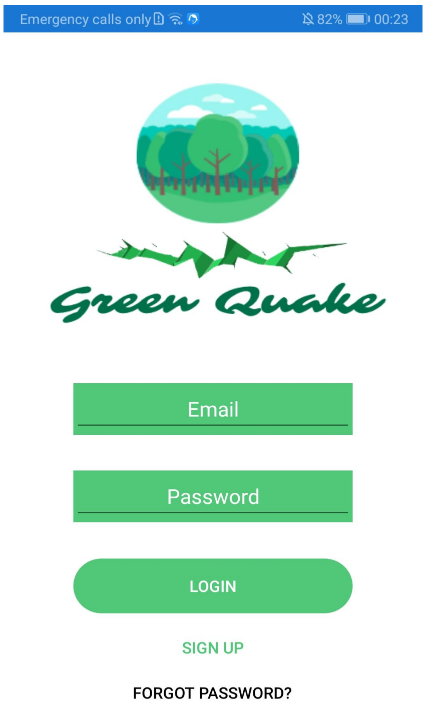
The application can be installed in two ways. The first way and the more complicated way is to clone the repository and open the solution using Visual Studio, connecting your device using a USB and selecting it from the drop down and clicking the play button.

The second and more simpler way to install this application is to just simply download the APK that I have provided in the project folder on your device and just simply clicking on it and installing it.

4 Application Screenshots

Below are listed all the application screenshots with short descriptions for them.

4.1 Login Screen

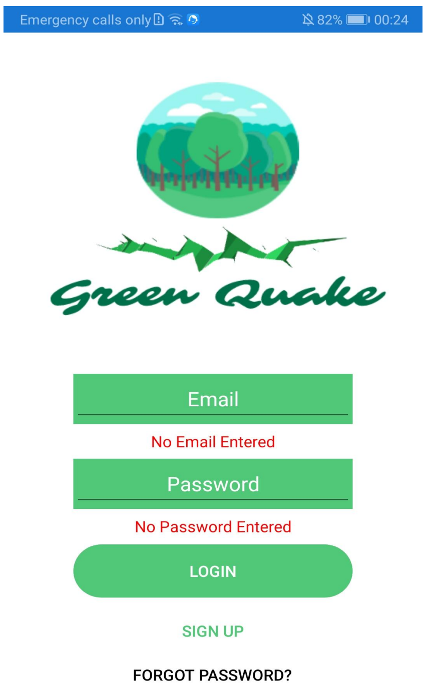


As you can see this is the **login** screen**.** This will be the opening screen for the application when the user is not signed in. A user can enter their email and password and click the login button to attempt to sign in or they can tap the Sign up option to navigate to the sign up page or they can tap on the Forgot Password? Option to navigate to the Forgot Password screen.

Upon successful login a nice splash screen appears which can not be documented as it is an animated event. After this splash screen the user is brought to the main menu.

Figure 1: Login Screen.

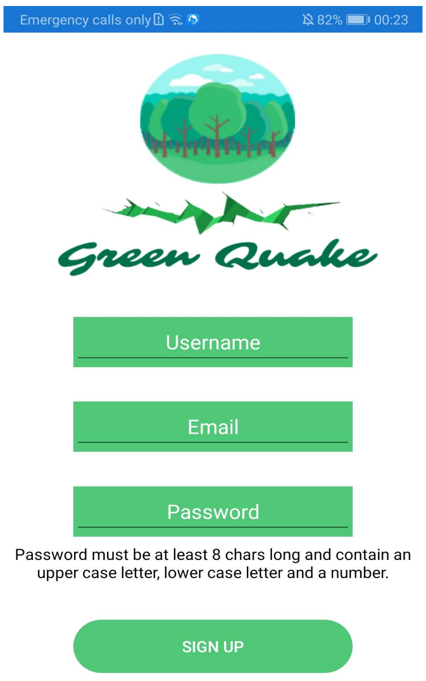
4.2 Login Screen With Error Messages



As you can see this is the **login** screen once more but now displays error messages for the invalid input fields. These error messages vary depending on the situation. If the user is not connected to the internet an alert pops up asking them to connect to the internet.

Figure 2: Login Screen With Error Messages.

4.3 Sign Up Screen

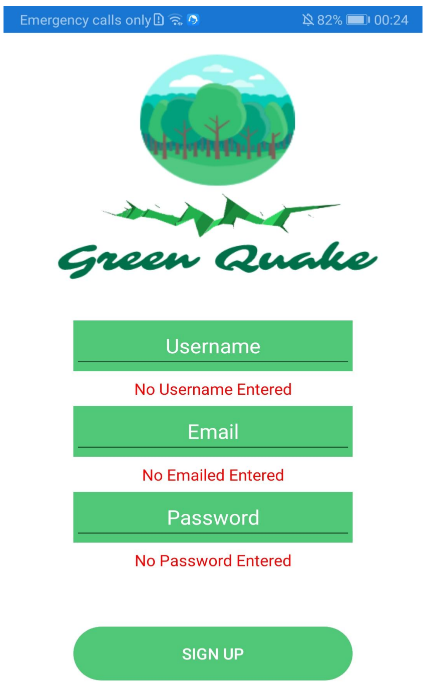


Password must be at least 8 chars long and contain an upper case letter, lower case letter and a number.

As you can see this is the **Sign Up** screen. A user can enter any username they wish and only a valid non duplicate email and a password which complies with the rules listed under the password fields. Upon successful signup an alert pops us saying your account has been created and redirects the user to the login page.

Figure 3: Sign Up Screen.

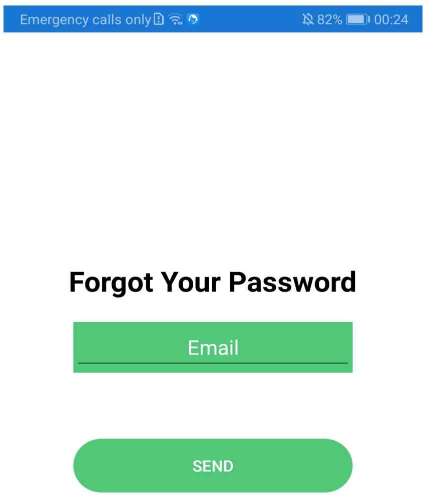
4.4 Sign Up Screen With Error Messages



As you can see this is the **Sign Up** screen once more but now displays error messages for the invalid input fields. These error messages vary depending on the situation. If the user is not connected to the internet an alert pops up asking them to connect to the internet. If the email already exists an alert pops up saying the email already exists and if there is an error a generic error alert pops up.

Figure 4: Sign Up Screen With Error Messages.

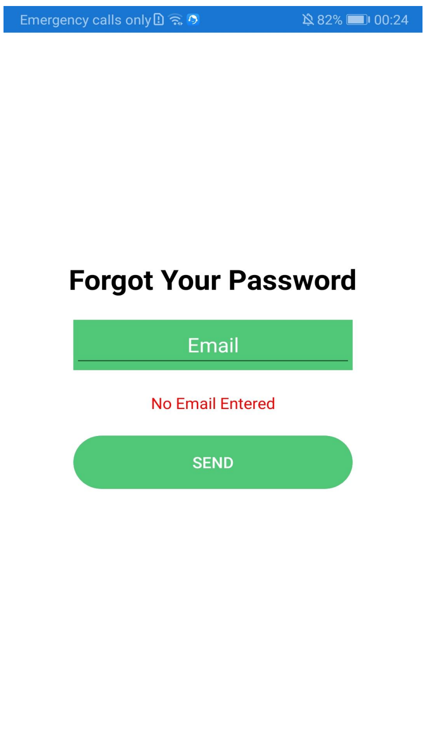
4.5 Forgot Password Screen



As you can see this is the **Forgot Password** screen. A user can enter their email and if the email exists a password reset email will be sent to that email address. An alert saying “If the email exists a password reset email has been sent to your email address”. Then the user is redirected to the login page.

Figure 5: Forgot Password Screen.

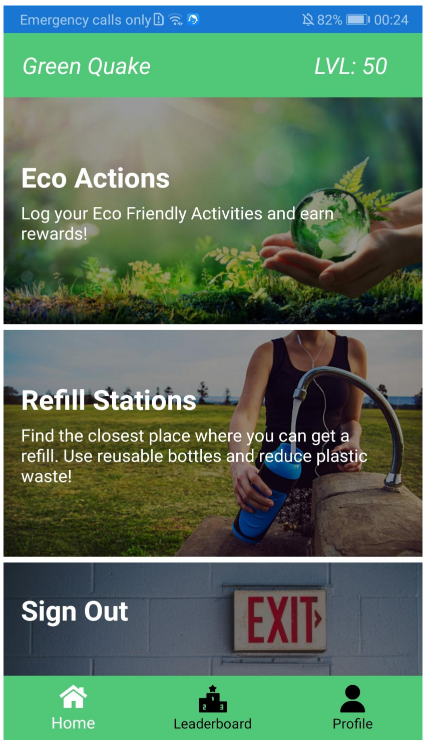
4.6 Forgot Password Screen With Error Messages



As you can see this is the **Forgot Password** screen once more but now displays an error message for the invalid input field. If the user is not connected to the internet or the email is invalid the corresponding alert box pops up.

Figure 6: Forgot Password Screen With Error Messages.

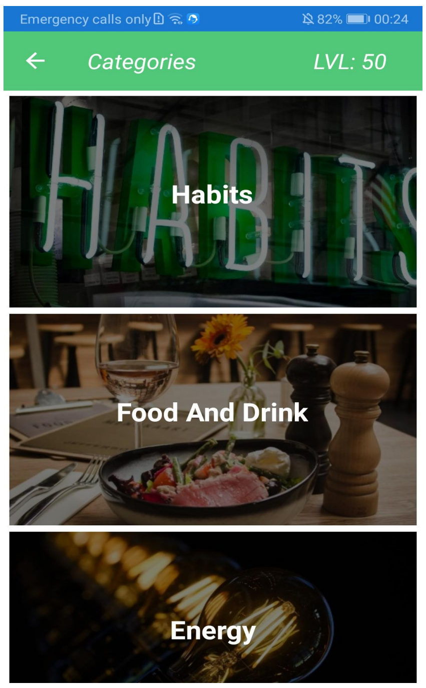
4.7 Main Menu Screen


Emergency calls only
82% 00:24
Green Quake
LVL: 50
Eco Actions
Log your Eco Friendly Activities and earn rewards!
Refill Stations
Find the closest place where you can get a refill. Use reusable bottles and reduce plastic waste!
Sign Out
EXIT
Home
Leaderboard
Profile

This is the **main menu** screen that appears after successful login. The user can navigate to the Eco Actions section by tapping on it and slo to the Refill Station section by tapping on it. In addition the user can navigate to the Leaderboard or Profile section using the navigation bar at the bottom of the page. The level of the user is displayed in the top right of the screen.

Figure 7: Main Menu Screen.

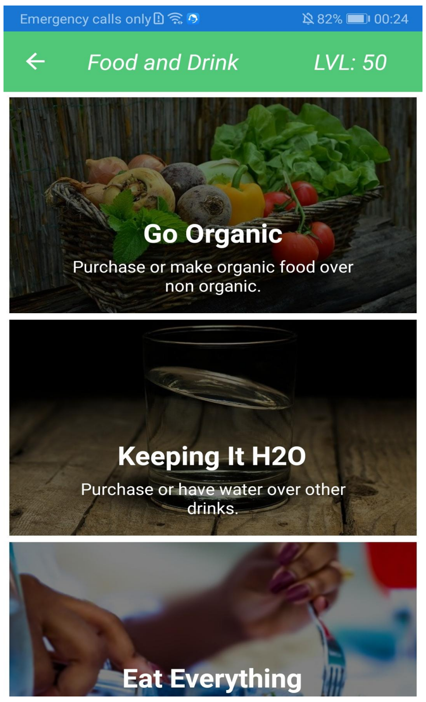
4.8 Categories Screen



This is the **Categories** screen that appears after a user selects Eco Actions from the previously mentioned main menu. From here the user can select one of 12 categories to log and eco action for. These categories are: Habits, Food And Drink, Energy, Travel, Shopping, Water, Home, Outdoors, Community, Waste, Work and Advanced categories.

Figure 8: Categories Screen.

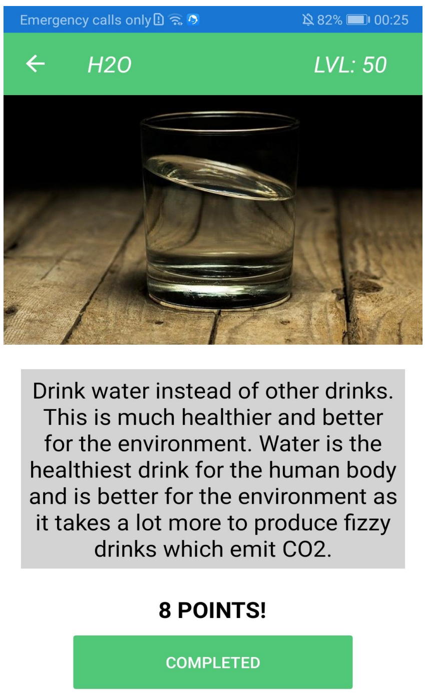
4.9 Food and Drink Subcategories Screen


Emergency calls only
82% 00:24
Food and Drink
LVL: 50
Go Organic
Purchase or make organic food over non organic.
Keeping It H2O
Purchase or have water over other drinks.
Eat Everything

This is the **Food and Drink** subcategories screen that appears after a user selects Food and Drink from the previously mentioned categories screen.

Figure 9: Food and Drink Subcategories Screen.

4.10 H20 Action Screen



Drink water instead of other drinks. This is much healthier and better for the environment. Water is the healthiest drink for the human body and is better for the environment as it takes a lot more to produce fizzy drinks which emit CO2.

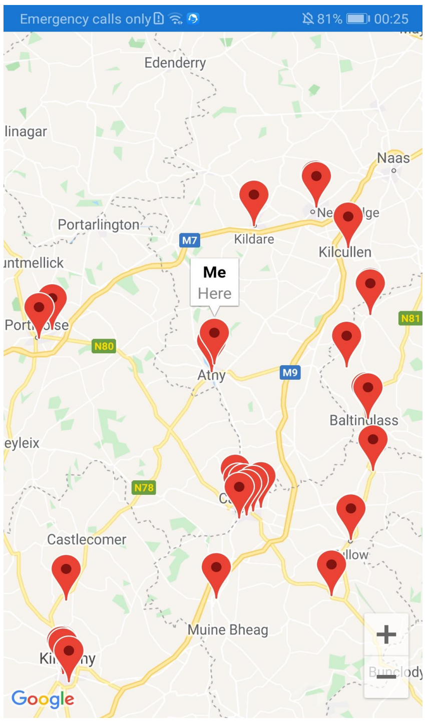
8 POINTS!

This is the **H20** action screen. This screen provides some details and interesting information about this particular action. The amount of points that it is worth is displayed below the text. The user can press the Completed button to log the action. Once this is pressed either the points get posted into the database and an alert box with the successful message is displayed to the user before redirecting them to the main menu or one of two error message alert boxes pop up.

The application does not allow more than 15 posts per day and an alert box saying this will be shown to the user and the post will not be submitted if the user tries to submit more than 15 actions per day. The app also only allows one submission per minute to prevent spamming and an alert box saying this also pops up for the user and does not submit the post.

Figure 10: H20 Action Screen.

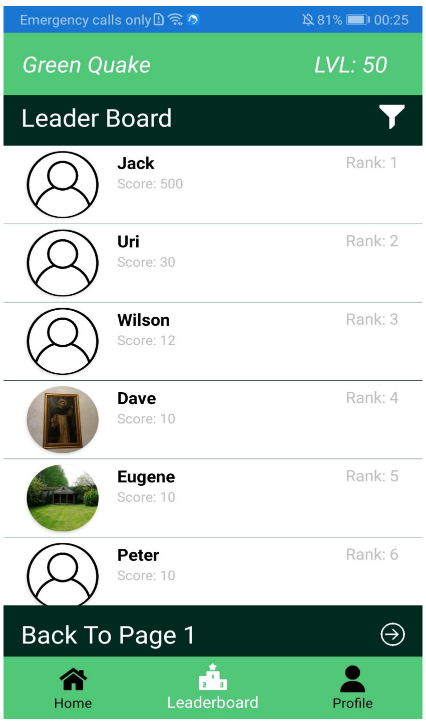
4.11 Refill Stations Screen



This is the **Refill Stations** screen that appears after a user selects Refill Station from the previously mentioned main menu. This screen loads a map at your current location and displays all the Refill Station on the map. The user can clearly see where the refill stations are located in retrospect to their location and can tap on each pin to view more details about it.

Figure 11: Refill Stations Screen.

4.12 Leaderboard Screen



Emergency calls only 81% 00:25

Green Quake LVL: 50

Leader Board

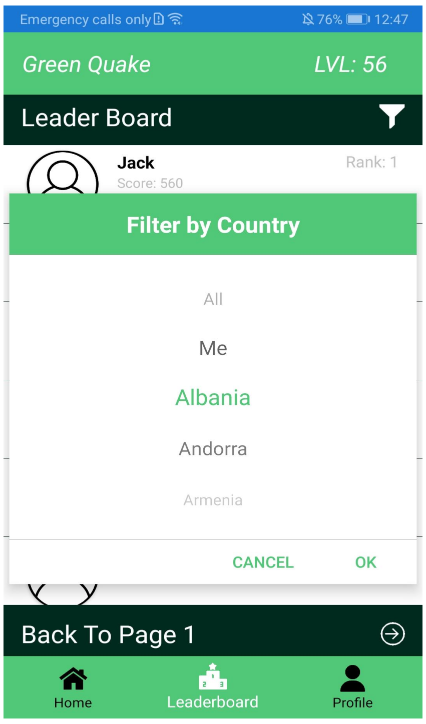
Jack	Score: 500	Rank: 1
Uri	Score: 30	Rank: 2
Wilson	Score: 12	Rank: 3
Dave	Score: 10	Rank: 4
Eugene	Score: 10	Rank: 5
Peter	Score: 10	Rank: 6

Back To Page 1

This is the **Leaderboard** screen that appears after a user selects the Leaderbaord from the navigation menu at the bottom of the screen on the previously mentioned main menu. On this screen the leaderboard displays 10 rows per page. The user can tap on a profile to view more information about the user. The user can also tap the icon in the top right to filter the leaderboard. The user can also tap the right arrow at the bottom right of the page to see the next 10 entries and tap the Back to Page on text to return to the first page. If there are no more pages to loads then an alert box with the appropriate message gets displayed.

Figure 12: Leaderboard Screen.

4.13 Leaderboard Screen When The Filter Button Is Tapped



Emergency calls only 76% 12:47

Green Quake LVL: 56

Leader Board

Jack Score: 560 Rank: 1

Filter by Country

All

Me

Albania

Andorra

Armenia

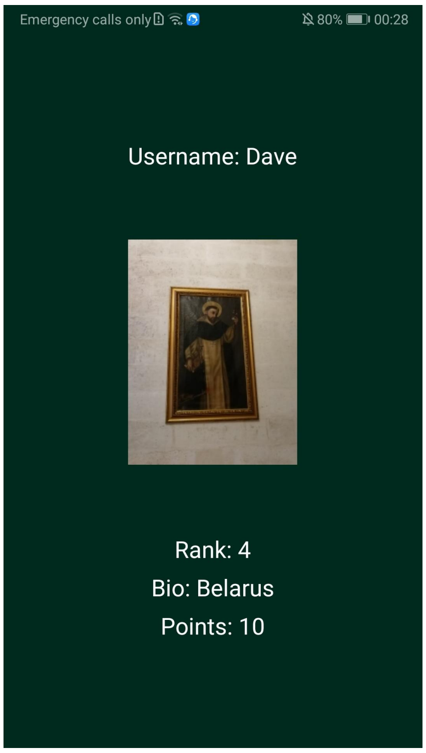
CANCEL OK

Back To Page 1

This is the **Leaderbaord** screen when the **filter icon** is tapped. A picker with the options All, Me and the list of nations appears. The user is allowed to filter to display all the users on the leaderboard, filter by nation or view their own position on the leaderboard global.

Figure 13: Leaderboard Screen When The Filter Button Is Tapped.

4.14 Leaderboard Pop Up Screen When A Profile Is Tapped On The Leaderboard



Username: Dave

Rank: 4

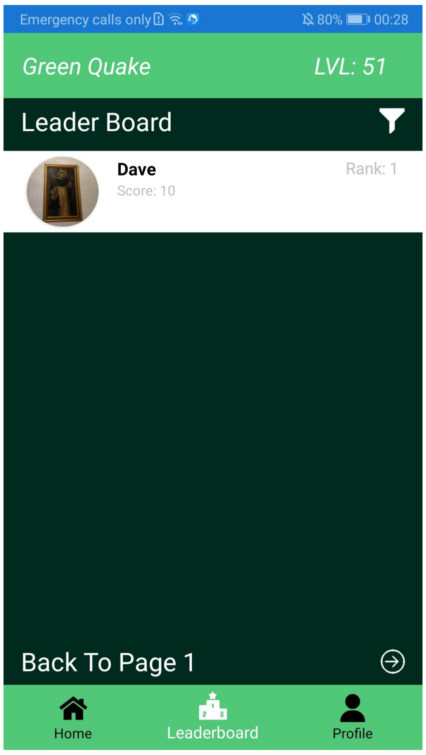
Bio: Belarus

Points: 10

This **popup** screen appears when a profile is tapped on from the leaderboard. It displays more information about the tapped on user. The information displayed is the username, the profile picture, the rank, the bio and the points.

Figure 14: Leaderboard Pop Up Screen When A Profile Is Tapped On The Leaderboard.

4.15 Filtered Leaderboard Screen



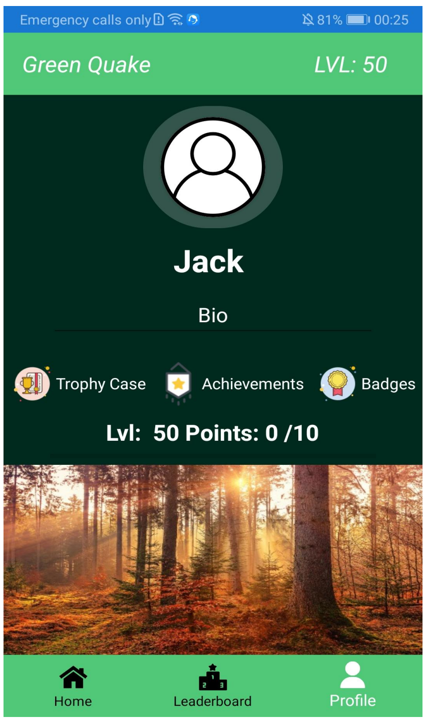
Dave Rank: 1

Score: 10

This is the **Leaderboard** after it has been **Filtered by Nation**. Only the profiles with the correct nation are displayed and they are ranked against other accounts of the same nation.

Figure 22: Filtered Leaderboard Screen.

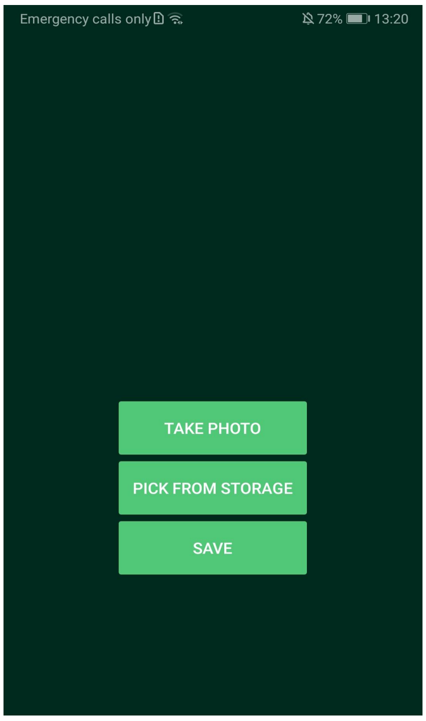
4.16 Profile Screen



This is the **Profile** screen that appears after a user selects the Profile from the navigation menu at the bottom of the screen on the previously mentioned main menu. The Profile screen has the profile picture and bio that the user can tap to change and their username. Below this are the Trophies, Achievements and Badges. By tapping on each of these the corresponding screen is displayed showing the Trophies, Achievements and Badges. Below this, the level of the user is displayed with a progress bar to the next level indicating how many points out of 10 the user needs to level up. Finally below this is the live avatar that is a mosaic that gets unlocked piece by piece on each level up. The mosaic gets unlocked after each level so each mosaic has 5 stages and there are 20 mosaics to unlock which makes them end at lvl 100.

Figure 16: Profile Screen.

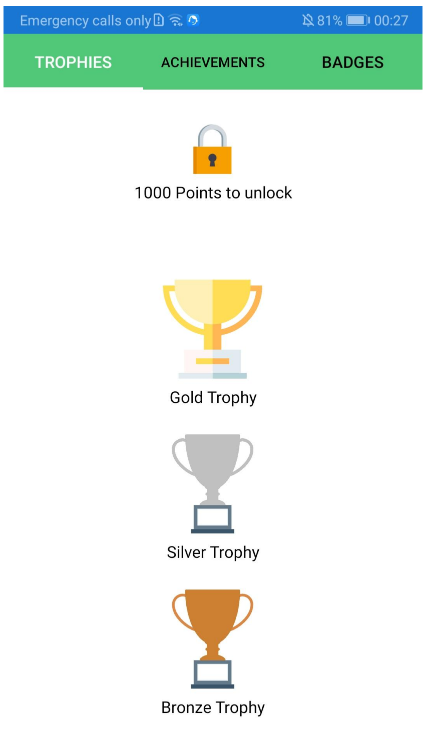
4.17 Popup Screen After The Profile Image Is Tapped



This **popup** screen appears when the user taps on the profile image on the profile screen. This popup allows the user to take a photo and save it as their profile picture or pick an image from the phone's storage and save it as their profile picture. Then the user is redirected to the profile screen.

Figure 17: Popup Screen After The Profile Image Is Tapped.

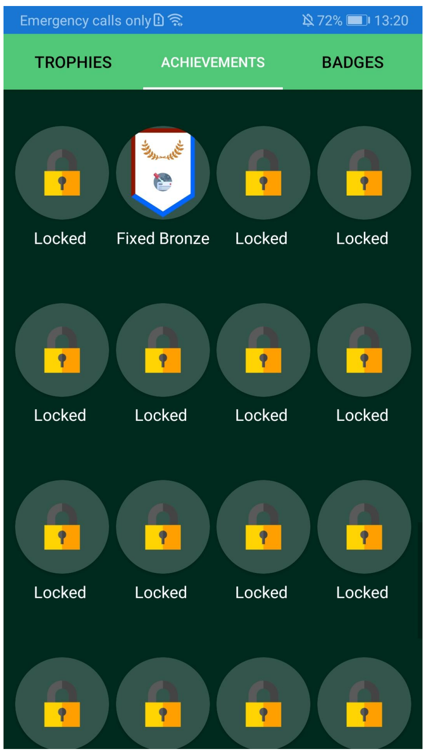
4.18 The Trophies Screen



This is the **Trophies** Screen that can be navigated to from the profile screen by tapping on the trophy case icon and text. The trophies are initially locked and get replaced with trophies as the requirements get met. Bronze trophy for 100 points, Silver for 250, Gold for 500 and Diamond for 1000.

Figure 18: The Trophies Screen.

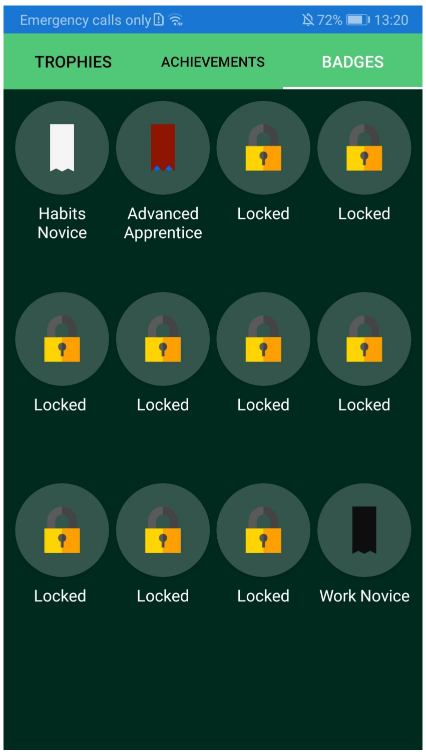
4.19 The Achievements Screen



This is the **Achievements** Screen that can be navigated to from the profile screen by tapping on the achievements icon and text. There are achievements for every action that can be logged which makes it over 80 achievements that the user can earn. In addition to this these are live achievements which can be bronze, silver and gold so there are over 260 achievements to earn in total. The achievements are initially locked and get unlocked by meeting the qualifying requirements. A bronze achievement is unlocked when a certain action gets logged 5 or more times, a silver achievement is unlocked when a certain action gets logged 15 or more times and finally a gold achievement is unlocked when a certain action gets logged 25 or more times.

Figure 19: The Achievements Screen.

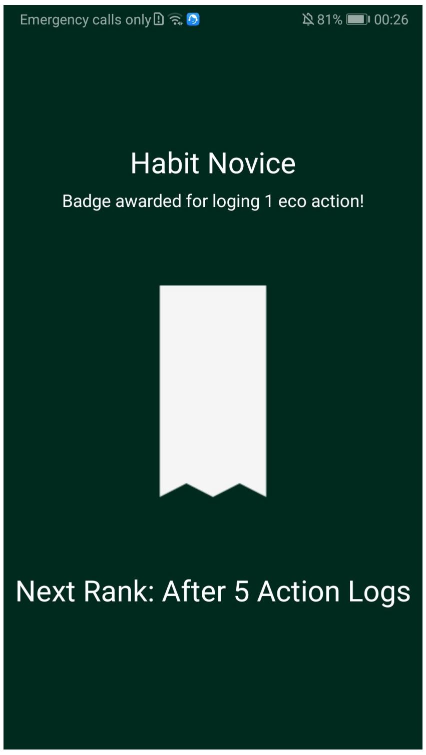
4.20 The BadgesScreen



This is the **Badges** Screen that can be navigated to from the profile screen by tapping on the badges icon and text. There are badges for every category that a user makes a login. There are 12 categories and there are 12 badges that can be unlocked and displayed. These are live badges and there are 6 levels in each category making it 72 badges that the user can earn. Each badge has a different design. The badges are initially locked and get unlocked when certain conditions are met. The Novice badge for a certain category gets unlocked when the user makes a single log or more in that category. Apprentice badge for 5 or more, Adept badge for 10 or more, Expert badge for 25 or more, Master badge for 50 or more and Legend Badge for 100 or more. All these badges can also be tapped on to get a close up and view more information.

Figure 20: The Badges Screen.

4.21 The Badge Popup Screen



Habit Novice

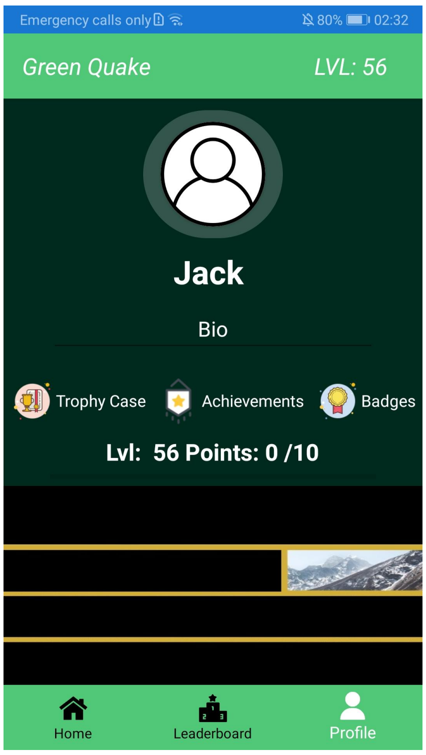
Badge awarded for loging 1 eco action!

Next Rank: After 5 Action Logs

This **Popup** screen appears when a user taps on the badges on the badges page. This image displays a close up of the badge and a description for the badge and what it was awarded for. It also tells the user the requirements to get the next badge.

Figure 21: The Badges Screen Popup When A Badge Is Tapped.

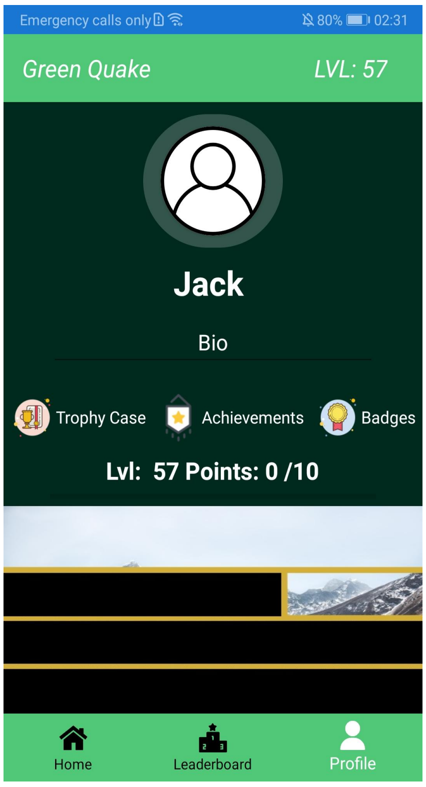
4.22 The Live Avatar Mosaics on Lvl 56



As you can see the mosaic only has one piece unlocked on level 56.

Figure 22: The Live Avatar Mosaics on Lvl 56.

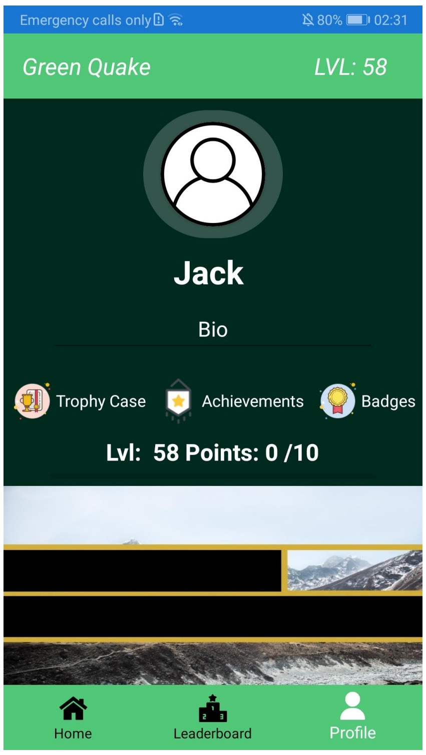
4.23 The Live Avatar Mosaics on Lvl 57



As you can see the mosaic only has two pieces unlocked on level 57.

Figure 23: The Live Avatar Mosaics on Lvl 57.

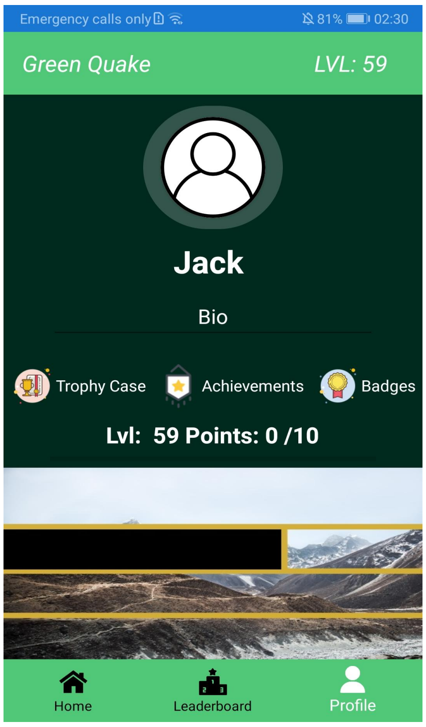
4.24 The Live Avatar Mosaics on Lvl 58



As you can see the mosaic has three pieces unlocked on level 58.

Figure 24: The Live Avatar Mosaics on Lvl 58.

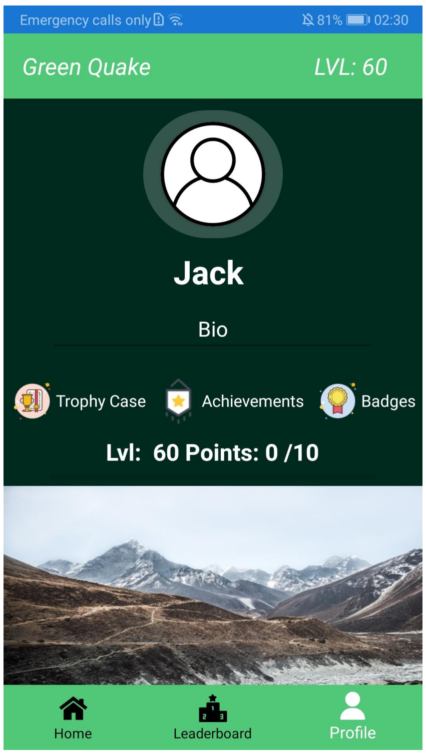
4.25 The Live Avatar Mosaics on Lvl 59



As you can see the mosaic has four pieces unlocked on level 59.

Figure 25: The Live Avatar Mosaics on Lvl 59.

4.26 The Live Avatar Mosaics on Lvl 60



As you can see the mosaic has all pieces unlocked on level 60. The process repeats itself every 5 levels with a new image each time all the way to level 100.

Figure 26: The Live Avatar Mosaics on Lvl 60.

5 Database

The Firebase Database is of a JSON format. Below is a visual representation of the Database used for this project

5.1 Firebase Database

```
application-green-quake-default-rtdb
-|AdvancedPoints
        -|UID
                -|fixCount
                -|numberOfLogs
                -|points
                -|username
-|EnergyPoints
        -|UID
                -|draftSealCount
                -|ductSealCount
                -|efficientThermostatCount
                -|fridgeCount
                -|fullDryerCount
                -|fullMachineCount
                -|hangDryCount
                -|insulateWaterCount
                -|isolateHomeCount
                -|ledLightBulbCount
                -|microwaveCount
                -|numberOfLogs
                -|offSocketCount
                -|points
                -|reBatteriesCount
                -|solarPanelCount
                -|username
```

```
-|FoodAndDrinkPoints
        -|UID
                -|eatAllCount
                -|foodDeliverCount
                -|noMeatCount
                -|numberOfLogs
                -|organicCount
                -|ownCoffeeCount
                -|points
                -|reCoffeeMugCount
                -|saveLeftOversCount
                -|steelStrawCount
                -|username
                -|waterOverFizzyCount
-|HabitsPoints
        -|UID
                -|brushingCount
                -|fullWasherCount
                -|matchesCount
                -|numberOfLogs
                -|offLigtsCount
                -|points
                -|showerCount
                -|timedShowerCount
                -|username
```

```
-|Points
        -|UID
                -|points
                -|username
-|SecuritySchecks
        -|UID
                -|counter
                -|date
                -|time


-|Station
        -|ID
                -|description
                -|label
                -|latitude
                -|longitude
-|Travel
        -|UID
                -|carpoolCount
                -|cycleCount
                -|ecoCarCount
                -|numberOfLogs
                -|points
                -|transportCount
                -|username
                -|walkCount
```

```
-|WastePoints
        -|UID
                -|billsCount
                -|bioBinBagsCount
                -|compostCount
                -|numberOfLogs
                -|points
                -|recyclingBinCount
                -|setUpRecyclingBinCount
                -|username
-|WorkPoints
        -|UID
                -|numberOfLogs
                -|offElectronicsCount
                -|paperCount
                -|points
                -|remoteWorkCount
                -|username
-|usernames
        -|username
                -|Uid
-|Users
        -|UID
                -|bio
                -|nation
                -|username
```

5.2 Firebase Storage

Firebase Storage is used to store images for users and it’s structure can be seen below:

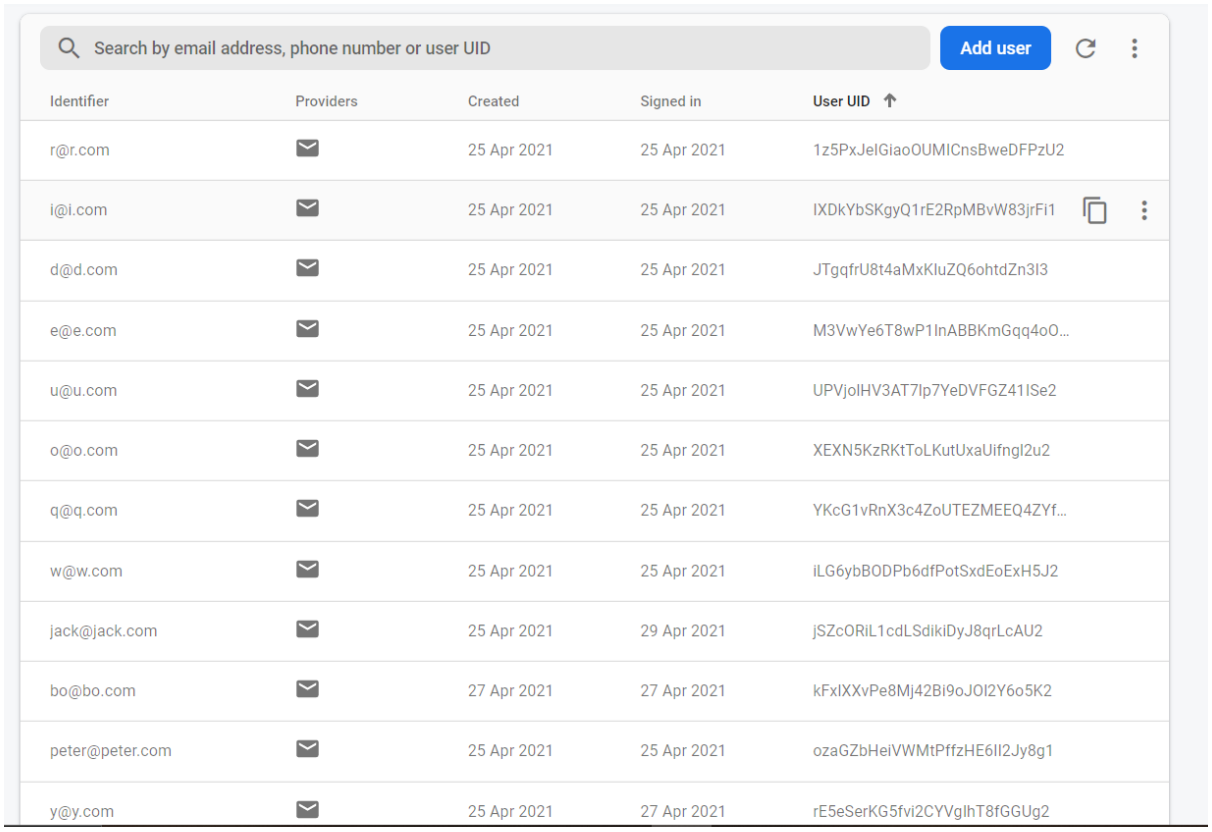
-|gs://application-green-quake.appspot.com

-|UID

-|filename

5.3 Firebase Authentication

The Firebase Authentication Database is shown below.



Identifier	Providers	Created	Signed in	User UID
r@r.com	✉	25 Apr 2021	25 Apr 2021	1z5PxJelGiaoOUMICnsBweDFPzU2
i@i.com	✉	25 Apr 2021	25 Apr 2021	IXDkYbSKgyQ1rE2RpMBvW83jrFi1
d@d.com	✉	25 Apr 2021	25 Apr 2021	JTgqfrU8t4aMxKluZQ6ohtdZn3I3
e@e.com	✉	25 Apr 2021	25 Apr 2021	M3VwYe6T8wP1InABBKmGqq4oO...
u@u.com	✉	25 Apr 2021	25 Apr 2021	UPVjolHV3AT7Ip7YeDVFGZ41ISe2
o@o.com	✉	25 Apr 2021	25 Apr 2021	XEXN5KzRKtToLKutUxaUifngl2u2
q@q.com	✉	25 Apr 2021	25 Apr 2021	YKcG1vRnX3c4ZoUTEZMEEQ4ZYf...
w@w.com	✉	25 Apr 2021	25 Apr 2021	iLG6ybBODPb6dfPotSxdEoExH5J2
jack@jack.com	✉	25 Apr 2021	29 Apr 2021	jSZcORiL1cdLSdikiDyJ8qrLcAU2
bo@bo.com	✉	27 Apr 2021	27 Apr 2021	kFxIXXvPe8Mj42Bi9oJOI2Y6o5K2
peter@peter.com	✉	25 Apr 2021	25 Apr 2021	ozaGZbHeiVWMtPffzHE6lI2Jy8g1
y@y.com	✉	25 Apr 2021	27 Apr 2021	rE5eSerKG5fvi2CYVglhT8fGGUg2

6 Code Listings

In this section the Code Listings are contained under their relevant heading and directory that is displayed above each class file.

```
/*! \class The AdvancedPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the AdvancedPoints Model Class
 *
 */
namespace Application_Green_Quake.Models
{
    class AdvancedPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int fixCount { get; set; }
    }
}
```

```csharp
/*! \class The AppConstants Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the AppConstants Model Class in contains the constants that are used throughout this application.
 *
 */
namespace Application_Green_Quake.Models
{
    public class AppConstants
    {
        //Int Constants
        public const int twoPoints = 2;
        public const int fourPoints = 4;
        public const int sixPoints = 6;
        public const int eightPoints = 8;
        public const int tenPoints = 10;

        //String Constants
        public const string googleMapsApiKey = "AIzaSyDf7Bq7gjei8Sp1AS_SWeapWyHe2rJtLmw";
        public const string twoPointsMsg = "2 Points Point Have been added";
        public const string fourPointsMsg = "4 Points Point Have been added";
        public const string sixPointsMsg = "6 Points Point Have been added";
        public const string eightPointsMsg = "8 Points Point Have been added";
        public const string tenPointsMsg = "10 Points Point Have been added";
    }
}
```

```
/*! \class The CommunityPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the CommunityPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class CommunityPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int createGroupCount { get; set; }
        public int communityCount { get; set; }
        public int donateCount { get; set; }
        public int groupCount { get; set; }
        public int shareCount{ get; set; }
        public int awarenessCount { get; set; }

    }
}
```

```
/*! \class The EnergyPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EnergyPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class EnergyPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int hangDryCount { get; set; }
        public int fullDryerCount { get; set; }
        public int insulateWaterCount { get; set; }
        public int efficientThermostatCount { get; set; }
        public int isolateHomeCount { get; set; }
        public int ledLightBulbCount { get; set; }
        public int fullMachineCount { get; set; }
        public int microwaveCount { get; set; }
        public int offSocketCount { get; set; }
        public int reBatteriesCount { get; set; }
        public int fridgeCount { get; set; }
        public int draftSealCount { get; set; }
        public int ductSealCount { get; set; }
        public int solarPanelCount { get; set; }
    }
}
```

```
/*! \class The FoodAndDrinkPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the FoodAndDrinkPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class FoodAndDrinkPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int organicCount { get; set; }
        public int eatAllCount { get; set; }
        public int foodDeliverCount { get; set; }
        public int noMeatCount { get; set; }
        public int ownCoffeeCount { get; set; }
        public int reCoffeeMugCount { get; set; }
        public int saveLeftOversCount { get; set; }
        public int steelStrawCount { get; set; }
        public int waterOverFizzyCount { get; set; }
    }
}
```

```
/*! \class The HabitsPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the HabitsPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class HabitsPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int brushingCount { get; set; }
        public int fullWasherCount { get; set; }
        public int showerCount { get; set; }
        public int timedShowerCount { get; set; }
        public int offLigtsCount { get; set; }
        public int matchesCount { get; set; }
    }
}
```

```
/*! \class The HomePoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the HomePoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class HomePoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int airOutCount { get; set; }
        public int nonHarmCount { get; set; }
        public int outsideCount { get; set; }
        public int plantIntoHomeCount { get; set; }
        public int toiletFlushCount { get; set; }
    }
}
```

```
/*! \class The ImageResourceExtension Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ImageResourceExtension Class. It is used for loading image files.
 *
 */
using System;
using System.Reflection;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Models
{
    [ContentProperty (nameof(Source))]
    class ImageResourceExtension : IMarkupExtension
    {
        public string Source { get; set; }

        public object ProvideValue(IServiceProvider serviceProvider)
        {
            //If the source is nothing then just null is returned
            if (Source == null)
            {
                return null;
            }

            var imageSource = ImageSource.FromResource(Source, typeof(ImageResourceExtension).GetTypeInfo().Assembly);

            return imageSource;

        }
    }
}
```

```
/*! \class The LeaderBoard Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the LeaderBoard Model Class.
 *
 */
using Xamarin.Forms;

namespace Application_Green_Quake.Models
{
    class LeaderBoard
    {
        public ImageSource image { get; set; }
        public string username { get; set; }
        public int points { get; set; }
        public string rank { get; set; }
        public string bio { get; set; }
        public string nation { get; set; }
        public string uid { get; set; }
    }
}
```

```
/*! \class The OutdoorsPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the OutdoorsPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class OutdoorsPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int campingCount { get; set; }
        public int picnicCount { get; set; }
        public int plantBushCount { get; set; }
        public int plantFlowerCount { get; set; }
        public int plantTreeCount { get; set; }
        public int scoopCount { get; set; }
        public int fruitGardenCount { get; set; }
        public int herbGardenCount { get; set; }
        public int vegetableGardenCount { get; set; }
        public int birdFeederCount { get; set; }

    }
}
```

```
/*! \class The Points Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Points Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class Points
    {
        public string username { get; set; }
        public int points { get; set; }

    }
}
```

```
/*! \class The SecurityChecks Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SecurityChecks Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class SecurityChecks
    {
        public string date { get; set; }
        public long time { get; set; }
        public int counter { get; set; }
    }
}
```

```
/*! \class The ShoppingPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ShoppingPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class ShoppingPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int clothNapkinCount { get; set; }
        public int clothTowelCount { get; set; }
        public int applianceCount { get; set; }
        public int productCount { get; set; }
        public int toothbrushCount { get; set; }
        public int clothesCount { get; set; }
        public int foodCount { get; set; }
        public int localCount { get; set; }
        public int looseLeafCount { get; set; }
        public int organicFoodCount { get; set; }
        public int reusableCount { get; set; }
        public int reBatCount { get; set; }
        public int reBagCount { get; set; }
    }
}
```

```csharp
/*! \class The Station Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Station Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class Station
    {
        public string description { get; set; }

        public string label { get; set; }

        public double latitude { get; set; }

        public double longitude { get; set; }
    }
}
```

```
/*! \class The TravelPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the TravelPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class TravelPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int carpoolCount { get; set; }
        public int cycleCount { get; set; }
        public int ecoCarCount { get; set; }
        public int transportCount { get; set; }
        public int walkCount { get; set; }
    }
}
```

```
/*! \class The Usernames Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Usernames Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class Usernames
    {
        public string Uid { get; set ; }
    }
}
```

```
/*! \class The Users Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Users Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class Users
    {
        public string username { get; set; }
        public string bio { get; set; }
        public string nation { get; set; }
    }
}
```

```
/*! \class The WastePoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WastePoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class WastePoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int billsCount { get; set; }
        public int compostCount { get; set; }
        public int setUpRecyclingBinCount { get; set; }
        public int bioBinBagsCount { get; set; }
        public int recyclingBinCount { get; set; }
    }
}
```

```
/*! \class The WaterPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WaterPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class WaterPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int cisternCount { get; set; }
        public int rainBarrelCount { get; set; }
        public int reWaterCount { get; set; }
        public int showerBucketCount { get; set; }
        public int wSShowerHeadCount { get; set; }
    }
}
```

```
/*! \class The WorkPoints Model Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WorkPoints Model Class.
 *
 */
namespace Application_Green_Quake.Models
{
    class WorkPoints
    {
        public string username { get; set; }
        public int points { get; set; }
        public int numberOfLogs { get; set; }
        public int paperCount { get; set; }
        public int offElectronicsCount { get; set; }
        public int remoteWorkCount { get; set; }
    }
}
```

```
/*! \class The AdvancedPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the AdvancedPointsUpdate ViewModel Class. It updates the data for the Advanced Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class AdvancedPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int fixCount2 = 0;
        string username = "";

        IAuth auth;
        /** This function updates the points in the Advanced category by ten points. It also increments the number of logs logged in the Advanced
          * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void FixPoints()
        {
            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();
            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("AdvancedPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<AdvancedPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("AdvancedPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<AdvancedPoints>()).numberOfLogs;

                numberOfLogs2++;

                fixCount2 = (await firebaseClient
                .Child("AdvancedPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<AdvancedPoints>()).fixCount;

                fixCount2++;

                await firebaseClient
                .Child("AdvancedPoints")
                .Child(auth.GetUid())
                .PutAsync(new AdvancedPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    fixCount = fixCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("AdvancedPoints")
                .Child(auth.GetUid())
                .PutAsync(new AdvancedPoints() { username = username, points = points2,
numberOfLogs = 1, fixCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("AdvancedPoints")
                .Child(auth.GetUid())
                .PutAsync(new AdvancedPoints() { username = username, points = points2,
numberOfLogs = 1, fixCount = 1 });
            }
        }
    }
}
```

```csharp
/*! \class The CommunityPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the CommunityPointsUpdate ViewModel Class. It updates the data for the Community Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class CommunityPointsUpdate
    {
        int points2 = 0;
        int createGroupCount2 = 0;
        int communityCount2 = 0;
        int donateCount2 = 0;
        int groupCount2 = 0;
        int shareCount2 = 0;
        int awarenessCount2 = 0;
        int numberOfLogs2 = 0;


        string username = "";

        IAuth auth;
        /** This function updates the points in the Community category by ten points. It also increments the number of logs logged in the Community
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void CreateGroupPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).numberOfLogs;

                numberOfLogs2++;

                createGroupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).createGroupCount;

                createGroupCount2++;

                communityCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).communityCount;

                donateCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).donateCount;

                groupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).groupCount;

                shareCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).shareCount;

                awarenessCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).awarenessCount;

                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    communityCount = communityCount2,
                    createGroupCount = createGroupCount2,
                    donateCount = donateCount2,
                    groupCount = groupCount2,
                    shareCount = shareCount2,
                    awarenessCount = awarenessCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;
                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, createGroupCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, createGroupCount = 1 });
            }
        }
        /** This function updates the points in the Community category by ten points. It 
also increments the number of logs logged in the Community
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void CommunityPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).numberOfLogs;

                numberOfLogs2++;

                createGroupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).createGroupCount;

                communityCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).communityCount;

                communityCount2++;

                donateCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).donateCount;

                groupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).groupCount;

                shareCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).shareCount;

                awarenessCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).awarenessCount;

                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    communityCount = communityCount2,
                    createGroupCount = createGroupCount2,
                    donateCount = donateCount2,
                    groupCount = groupCount2,
                    shareCount = shareCount2,
                    awarenessCount = awarenessCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, communityCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, communityCount = 1 });
            }
        }
        /** This function updates the points in the Community category by ten points. It 
also increments the number of logs logged in the Community
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void DonatePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).numberOfLogs;

                numberOfLogs2++;

                createGroupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).createGroupCount;

                communityCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).communityCount;

                donateCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).donateCount;

                donateCount2++;

                groupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).groupCount;

                shareCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).shareCount;

                awarenessCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).awarenessCount;

                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    communityCount = communityCount2,
                    createGroupCount = createGroupCount2,
                    donateCount = donateCount2,
                    groupCount = groupCount2,
                    shareCount = shareCount2,
                    awarenessCount = awarenessCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, donateCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, donateCount = 1 });
            }
```

```
        }
        /** This function updates the points in the Community category by eight points. It also increments the number of logs logged in the Community
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void GroupPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).numberOfLogs;

                numberOfLogs2++;

                createGroupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).createGroupCount;

                communityCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).communityCount;

                donateCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).donateCount;

                 groupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).groupCount;

                groupCount2++;

                shareCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).shareCount;

                awarenessCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).awarenessCount;

                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    communityCount = communityCount2,
                    createGroupCount = createGroupCount2,
                    donateCount = donateCount2,
                    groupCount = groupCount2,
                    shareCount = shareCount2,
                    awarenessCount = awarenessCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, groupCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, groupCount = 1 });
            }
        }
        /** This function updates the points in the Community category by eight points. It
also increments the number of logs logged in the Community
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void SharePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).numberOfLogs;

                numberOfLogs2++;

                createGroupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).createGroupCount;

                communityCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).communityCount;

                donateCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).donateCount;

                groupCount2 = (await firebaseClient
               .Child("CommunityPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<CommunityPoints>()).groupCount;

                shareCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).shareCount;

                shareCount2++;

                awarenessCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).awarenessCount;

                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    communityCount = communityCount2,
                    createGroupCount = createGroupCount2,
                    donateCount = donateCount2,
                    groupCount = groupCount2,
                    shareCount = shareCount2,
                    awarenessCount = awarenessCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, shareCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, shareCount = 1 });
            }
        }
        /** This function updates the points in the Community category by eight points. It 
also increments the number of logs logged in the Community
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void awarenessPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).numberOfLogs;

                numberOfLogs2++;

                createGroupCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).createGroupCount;

                communityCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).communityCount;

                donateCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).donateCount;

                groupCount2 = (await firebaseClient
               .Child("CommunityPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<CommunityPoints>()).groupCount;

                shareCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).shareCount;

                awarenessCount2 = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).awarenessCount;

                awarenessCount2++;

                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    communityCount = communityCount2,
                    createGroupCount = createGroupCount2,
                    donateCount = donateCount2,
                    groupCount = groupCount2,
                    shareCount = shareCount2,
                    awarenessCount = awarenessCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, awarenessCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .PutAsync(new CommunityPoints() { username = username, points = points2, 
numberOfLogs = 1, awarenessCount = 1 });
            }
        }
    }
}
```

```
/*! \class The EnergyPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EnergyPointsUpdate ViewModel Class. It updates the data for the Energy Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class EnergyPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int hangDry2 = 0;
        int fullDryerCount2 = 0;
        int draftSealCount2 = 0;
        int ductSealCount2 = 0;
        int efficientThermostatCount2 = 0;
        int fridgeCount2 = 0;
        int fullMachineCount2 = 0;
        int insulateWaterCount2 = 0;
        int isolateHomeCount2 = 0;
        int ledLightBulbCount2 = 0;
        int microwaveCount2 = 0;
        int solarPanelCount2 = 0;
        int offSocketCount2 = 0;
        int reBatteriesCount2 = 0;


        string username = "";

        IAuth auth;
        /** This function updates the points in the Energy category by ten points. It also increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void HangDryPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                hangDry2++;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                     username = username,
                     points = points2,
                     numberOfLogs = numberOfLogs2,
                     hangDryCount = hangDry2,
                     draftSealCount = draftSealCount2,
                     ductSealCount = ductSealCount2,
                     efficientThermostatCount = efficientThermostatCount2,
                     fridgeCount = fridgeCount2,
                     fullDryerCount = fullDryerCount2,
                     insulateWaterCount = insulateWaterCount2,
                     isolateHomeCount = isolateHomeCount2,
                     ledLightBulbCount =ledLightBulbCount2,
                     microwaveCount = microwaveCount2,
                     offSocketCount = offSocketCount2,
                     reBatteriesCount = reBatteriesCount2,
                     solarPanelCount = solarPanelCount2,
                     fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, hangDryCount = 1});;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, hangDryCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by ten points. It also 
increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void DryerFullPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullDryerCount2++;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, fullDryerCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, fullDryerCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by eight points. It 
also increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void EfficientThermostatPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                efficientThermostatCount2++;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, efficientThermostatCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, efficientThermostatCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by ten points. It also 
increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void InsulateWaterPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                insulateWaterCount2++;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, insulateWaterCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, insulateWaterCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by ten points. It also
increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void IsolateHomePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                isolateHomeCount2++;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, isolateHomeCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, isolateHomeCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by ten points. It also
increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void LedLightsPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                ledLightBulbCount2++;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, ledLightBulbCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, ledLightBulbCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by eight points. It 
also increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void MachineFullPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                fullMachineCount2++;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, fullMachineCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, fullMachineCount = 1 });
            }
        }

        public async void MicrowavePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                microwaveCount2++;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, microwaveCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, microwaveCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by four points. It 
also increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void SocketPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                offSocketCount2++;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, offSocketCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, offSocketCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by six points. It also
increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
```

```
        */
        public async void ReBatteriesPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                reBatteriesCount2++;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, reBatteriesCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, reBatteriesCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by eight points. It
also increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void FridgePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fridgeCount2++;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, fridgeCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, fridgeCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by ten points. It also
increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void SealDraftsPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                draftSealCount2++;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, draftSealCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, draftSealCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by eight points. It
also increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void SealDuctsPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                ductSealCount2++;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, ductSealCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2, 
numberOfLogs = 1, ductSealCount = 1 });
            }
        }
        /** This function updates the points in the Energy category by ten points. It also 
increments the number of logs logged in the Energy
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void SolarPanelPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;

                numberOfLogs2++;

                hangDry2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).hangDryCount;

                draftSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).draftSealCount;

                ductSealCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ductSealCount;

                efficientThermostatCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;

                fridgeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fridgeCount;

                fullDryerCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullDryerCount;

                fullMachineCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).fullMachineCount;

                insulateWaterCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;

                ledLightBulbCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;

                microwaveCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).microwaveCount;

                offSocketCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).offSocketCount;

                reBatteriesCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;

                solarPanelCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).solarPanelCount;

                solarPanelCount2++;

                isolateHomeCount2 = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;

                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    hangDryCount = hangDry2,
                    draftSealCount = draftSealCount2,
                    ductSealCount = ductSealCount2,
                    efficientThermostatCount = efficientThermostatCount2,
                    fridgeCount = fridgeCount2,
                    fullDryerCount = fullDryerCount2,
                    insulateWaterCount = insulateWaterCount2,
                    isolateHomeCount = isolateHomeCount2,
                    ledLightBulbCount = ledLightBulbCount2,
                    microwaveCount = microwaveCount2,
                    offSocketCount = offSocketCount2,
                    reBatteriesCount = reBatteriesCount2,
                    solarPanelCount = solarPanelCount2,
                    fullMachineCount = fullMachineCount2
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, solarPanelCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, solarPanelCount = 1 });
            }
        }
    }
}
```

```
/*! \class The FoodAndDrinkPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the FoodAndDrinkPointsUpdate ViewModel Class. It updates the data for the Food And Drink Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class FoodAndDrinkPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int organicCount2 = 0;
        int eatAllCount2 = 0;
        int foodDeliverCount2 = 0;
        int noMeatCount2 = 0;
        int ownCoffeeCount2 = 0;
        int reCoffeeMugCount2 = 0;
        int saveLeftOversCount2 = 0;
        int steelStrawCount2 = 0;
        int waterOverFizzyCount2 = 0;

        string username = "";

        IAuth auth;
        /** This function updates the points in the FoodAndDrink category by eight points. It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void OrganicPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                organicCount2++;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, organicCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, organicCount = 1 });
            }
        }
        /** This function updates the points in the FoodAndDrink category by four points. 
It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void EatAllPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                eatAllCount2++;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, eatAllCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, eatAllCount = 1 });
            }
        }
        /** This function updates the points in the FoodAndDrink category by six points. 
It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void FoodDelivredPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                foodDeliverCount2++;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, foodDeliverCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, foodDeliverCount = 1 });
            }
        }
        /** This function updates the points in the FoodAndDrink category by ten points.
It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void NoMeatPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                noMeatCount2++;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, noMeatCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, noMeatCount = 1 });
            }
        }
        /** This function updates the points in the FoodAndDrink category by two points.
```

```
It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void OwnCoffeePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                ownCoffeeCount2++;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, ownCoffeeCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, ownCoffeeCount = 1 });
            }
        }
        /** This function updates the points in the FoodAndDrink category by four points. 
It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void ReCoffeeMugPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                reCoffeeMugCount2++;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, reCoffeeMugCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, reCoffeeMugCount = 1 });
            }
        }
        /** This function updates the points in the FoodAndDrink category by six points. 
It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void SaveLeftOversPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                saveLeftOversCount2++;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, saveLeftOversCount = 1 }); ;
```

```

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, saveLeftOversCount = 1 });
            }
        }
        /** This function updates the points in the FoodAndDrink category by four points. 
It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void SteelStrawPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                steelStrawCount2++;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, steelStrawCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, steelStrawCount = 1 });
            }
        }
        /** This function updates the points in the FoodAndDrink category by six points. 
It also increments the number of logs logged in the FoodAndDrink
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void WaterOverFizzyPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;

                numberOfLogs2++;

                organicCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;

                eatAllCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;

                foodDeliverCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;

                noMeatCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;

                ownCoffeeCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;

                reCoffeeMugCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;

                saveLeftOversCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;

                steelStrawCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;

                waterOverFizzyCount2 = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;

                waterOverFizzyCount2++;

                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    organicCount = organicCount2,
                    eatAllCount = eatAllCount2,
                    foodDeliverCount = foodDeliverCount2,
                    noMeatCount = noMeatCount2,
                    ownCoffeeCount = ownCoffeeCount2,
                    reCoffeeMugCount = reCoffeeMugCount2,
                    saveLeftOversCount = saveLeftOversCount2,
                    steelStrawCount = steelStrawCount2,
                    waterOverFizzyCount = waterOverFizzyCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, waterOverFizzyCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .PutAsync(new FoodAndDrinkPoints() { username = username, points = 
points2, numberOfLogs = 1, waterOverFizzyCount = 1 });
            }
        }
    }
}
```

```
/*! \class The GetAchievementsData ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the GetAchievementsData ViewModel Class. It gets data that is needed for Achievements from firebase.
 */
using System;
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class GetAchievementsData
    {
        IAuth auth;
        public static int fixCount = 0;
        public static int wSShowerHeadCount = 0;
        public static int waterOverFizzyCount = 0;
        public static int createGroupCount = 0;
        public static int communityCount = 0;
        public static int donateCount = 0;
        public static int groupCount = 0;
        public static int shareCount = 0;
        public static int awarenessCount = 0;
        public static int fullDryerCount = 0;
        public static int insulateWaterCount = 0;
        public static int efficientThermostatCount = 0;
        public static int isolateHomeCount = 0;
        public static int ledLightBulbCount = 0;
        public static int microwaveCount = 0;
        public static int offSocketCount = 0;
        public static int reBatteriesCount = 0;
        public static int reBatCount = 0;
        public static int fridgeCount = 0;
        public static int draftSealCount = 0;
        public static int ductSealCount = 0;
        public static int solarPanelCount = 0;
        public static int eatAllCount = 0;
        public static int foodDeliverCount = 0;
        public static int noMeatCount = 0;
        public static int ownCoffeeCount = 0;
        public static int reCoffeeMugCount = 0;
        public static int saveLeftOversCount = 0;
        public static int steelStrawCount = 0;
        public static int brushingCount = 0;
        public static int fullWasherCount = 0;
        public static int showerCount = 0;
        public static int timedShowerCount = 0;
        public static int offLigtsCount = 0;
        public static int matchesCount = 0;
        public static int airOutCount = 0;
        public static int nonHarmCount = 0;
        public static int outsideCount = 0;
        public static int plantIntoHomeCount = 0;
        public static int toiletFlushCount = 0;
        public static int campingCount = 0;
        public static int picnicCount = 0;
        public static int plantBushCount = 0;
        public static int plantFlowerCount = 0;
        public static int plantTreeCount = 0;
        public static int scoopCount = 0;
        public static int fruitGardenCount = 0;
        public static int herbGardenCount = 0;
        public static int vegetableGardenCount = 0;
        public static int birdFeederCount = 0;
        public static int clothNapkinCount = 0;
        public static int clothTowelCount = 0;
        public static int applianceCount = 0;
        public static int productCount = 0;
        public static int toothbrushCount = 0;
        public static int clothesCount = 0;
        public static int localCount = 0;
        public static int looseLeafCount = 0;
        public static int organicFoodCount = 0;
        public static int reusableCount = 0;
        public static int reBagCount = 0;
        public static int carpoolCount = 0;
        public static int cycleCount = 0;
        public static int ecoCarCount = 0;
        public static int transportCount = 0;
        public static int walkCount = 0;
        public static int billsCount = 0;
        public static int compostCount = 0;
        public static int setUpRecyclingBinCount = 0;
        public static int bioBinBagsCount = 0;
        public static int recyclingBinCount = 0;
        public static int cisternCount = 0;
        public static int rainBarrelCount = 0;
        public static int reWaterCount = 0;
        public static int showerBucketCount = 0;
        public static int paperCount = 0;
        public static int offElectronicsCount = 0;
        public static int remoteWorkCount = 0;
        public static int hangDryCount = 0;
        public static int foodCount = 0;

        /**
         * This function sets the data that is needed for the Achievements Screen
        */
        public async void SetAchievementsData()
        {
            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                fixCount = (await firebaseClient
               .Child("AdvancedPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<AdvancedPoints>()).fixCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                createGroupCount = (await firebaseClient
               .Child("CommunityPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<CommunityPoints>()).createGroupCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                communityCount = (await firebaseClient
               .Child("CommunityPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<CommunityPoints>()).communityCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                donateCount = (await firebaseClient
               .Child("CommunityPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<CommunityPoints>()).donateCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                groupCount = (await firebaseClient
               .Child("CommunityPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<CommunityPoints>()).groupCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                shareCount = (await firebaseClient
               .Child("CommunityPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<CommunityPoints>()).shareCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                awarenessCount = (await firebaseClient
               .Child("CommunityPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<CommunityPoints>()).awarenessCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                hangDryCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).hangDryCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                fullDryerCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                insulateWaterCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                efficientThermostatCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                isolateHomeCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                ledLightBulbCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                microwaveCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).microwaveCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                offSocketCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).offSocketCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                reBatteriesCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                reBatCount = (await firebaseClient
              .Child("ShoppingPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<ShoppingPoints>()).reBatCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                fridgeCount = (await firebaseClient
              .Child("EnergyPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<EnergyPoints>()).fridgeCount;
            }
            catch (Exception e)
```

```
            {
                Console.Write(e);
            }
            try
            {
                draftSealCount = (await firebaseClient
               .Child("EnergyPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<EnergyPoints>()).draftSealCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }

            try
            {
                ductSealCount = (await firebaseClient
               .Child("EnergyPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<EnergyPoints>()).ductSealCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                solarPanelCount = (await firebaseClient
               .Child("EnergyPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                eatAllCount = (await firebaseClient
               .Child("FoodAndDrinkPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                foodDeliverCount = (await firebaseClient
               .Child("FoodAndDrinkPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                noMeatCount = (await firebaseClient
              .Child("FoodAndDrinkPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                ownCoffeeCount = (await firebaseClient
              .Child("FoodAndDrinkPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                reCoffeeMugCount = (await firebaseClient
              .Child("FoodAndDrinkPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                saveLeftOversCount = (await firebaseClient
              .Child("FoodAndDrinkPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                steelStrawCount = (await firebaseClient
              .Child("FoodAndDrinkPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                waterOverFizzyCount = (await firebaseClient
              .Child("FoodAndDrinkPoints")
              .Child(auth.GetUid())
              .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                brushingCount = (await firebaseClient
               .Child("HabitsPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<HabitsPoints>()).brushingCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                fullWasherCount = (await firebaseClient
               .Child("HabitsPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                showerCount = (await firebaseClient
               .Child("HabitsPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<HabitsPoints>()).showerCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                timedShowerCount = (await firebaseClient
               .Child("HabitsPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                offLigtsCount = (await firebaseClient
               .Child("HabitsPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                matchesCount = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).matchesCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                airOutCount = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).airOutCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                nonHarmCount = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).nonHarmCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                outsideCount = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).outsideCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                plantIntoHomeCount = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                toiletFlushCount = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).toiletFlushCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                campingCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                picnicCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                plantBushCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                plantFlowerCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                plantTreeCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                scoopCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                fruitGardenCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                herbGardenCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                vegetableGardenCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                birdFeederCount = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                clothNapkinCount = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                clothTowelCount = (await firebaseClient
               .Child("ShoppingPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                applianceCount = (await firebaseClient
               .Child("ShoppingPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<ShoppingPoints>()).applianceCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                productCount = (await firebaseClient
               .Child("ShoppingPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<ShoppingPoints>()).productCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                toothbrushCount = (await firebaseClient
               .Child("ShoppingPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                clothesCount = (await firebaseClient
               .Child("ShoppingPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<ShoppingPoints>()).clothesCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                localCount = (await firebaseClient
               .Child("ShoppingPoints")
               .Child(auth.GetUid())
```

```
                .OnceSingleAsync<ShoppingPoints>()).localCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                looseLeafCount = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                organicFoodCount = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                reusableCount = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reusableCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                reBagCount = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBagCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                foodCount = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).foodCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                carpoolCount = (await firebaseClient
               .Child("TravelPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<TravelPoints>()).carpoolCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                cycleCount = (await firebaseClient
               .Child("TravelPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<TravelPoints>()).cycleCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                ecoCarCount = (await firebaseClient
               .Child("TravelPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<TravelPoints>()).ecoCarCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                transportCount = (await firebaseClient
               .Child("TravelPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<TravelPoints>()).transportCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                walkCount = (await firebaseClient
               .Child("TravelPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<TravelPoints>()).walkCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                billsCount = (await firebaseClient
               .Child("WastePoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WastePoints>()).billsCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                compostCount = (await firebaseClient
               .Child("WastePoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WastePoints>()).compostCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                setUpRecyclingBinCount = (await firebaseClient
               .Child("WastePoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                bioBinBagsCount = (await firebaseClient
               .Child("WastePoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                recyclingBinCount = (await firebaseClient
               .Child("WastePoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WastePoints>()).recyclingBinCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                cisternCount = (await firebaseClient
               .Child("WaterPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WaterPoints>()).cisternCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                rainBarrelCount = (await firebaseClient
               .Child("WaterPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                reWaterCount = (await firebaseClient
               .Child("WaterPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WaterPoints>()).reWaterCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                showerBucketCount = (await firebaseClient
               .Child("WaterPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WaterPoints>()).showerBucketCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                wSShowerHeadCount = (await firebaseClient
               .Child("WaterPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                paperCount = (await firebaseClient
               .Child("WorkPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WorkPoints>()).paperCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                offElectronicsCount = (await firebaseClient
               .Child("WorkPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WorkPoints>()).offElectronicsCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                remoteWorkCount = (await firebaseClient
               .Child("WorkPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<WorkPoints>()).remoteWorkCount;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
        }
    }
}
```

```
/*! \class The GetBadgeData ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the GetBadgeData ViewModel Class. It gets data that is needed for badges from firebase.
 */
using System;
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class GetBadgeData
    {
        IAuth auth;
        public static int advancedLog = 0;
        public static int habitsLog = 0;
        public static int communityLog = 0;
        public static int energyLog = 0;
        public static int foodDrinkLog = 0;
        public static int homeLog = 0;
        public static int outdoorsLog = 0;
        public static int shoppingLog = 0;
        public static int travelLog = 0;
        public static int wasteLog = 0;
        public static int waterLog = 0;
        public static int workLog = 0;

        /**
         * This function sets the data that is needed for the badges screen
        */
        public async void SetBadgeData()
        {
            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                advancedLog = (await firebaseClient
                .Child("AdvancedPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<AdvancedPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                habitsLog = (await firebaseClient
               .Child("HabitsPoints")
               .Child(auth.GetUid())
               .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                communityLog = (await firebaseClient
                .Child("CommunityPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                energyLog = (await firebaseClient
                .Child("EnergyPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                foodDrinkLog = (await firebaseClient
                .Child("FoodAndDrinkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                homeLog = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                outdoorsLog = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                shoppingLog = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                travelLog = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                wasteLog = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                waterLog = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
            try
            {
                workLog = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).numberOfLogs;
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
        }
    }
}
```

```
/*! \class The GetData ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the GetData ViewModel Class. It gets data that is needed for the applications back end and front end.
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class GetData
    {
        IAuth auth;
        public static string username = "";
        public static string bio = "";
        public static int points = 0;
        public static int lvl = 0;
        public static string nation = "";

        /**
         * This function sets the data to be used for the font end in this application. It sets the username, bio, points, nation and lvl for the logged in
         * user. These are saved in global public variables that are used across the application for front end.
        */
        public async void SetData()
        {
            try
            {
                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                bio = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).bio;

                nation = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).nation;

                points = (await firebaseClient
                    .Child("Points")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Points>()).points;

                lvl = (int) Math.Floor((float) points / 10);
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
        }
        /**
         * This function just sets the level of the user in a global static variable shared 
across all the classes to be displayed for the front end.
        */
        public async void SetLvl()
        {
            try
            {
                FirebaseClient firebaseClient =
                    new FirebaseClient("https://application-green-quake-default-
rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                points = (await firebaseClient
                    .Child("Points")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Points>()).points;

                lvl = (int) Math.Floor((float) points / 10);
            }
            catch (Exception e)
            {
                Console.Write(e);
            }
        }
    }
}
```

```
/*! \class The HabitsPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the HabitsPointsUpdate ViewModel Class. It updates the data for the
Habits Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending
this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class HabitsPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int brushingCount2 = 0;
        int fullWasherCount2 = 0;
        int showerCount2 = 0;
        int timedShowerCount2 = 0;
        int offLigtsCount2 = 0;
        int matchesCount2 = 0;

        string username = "";

        IAuth auth;
        /** This function updates the points in the Habits category by two points. It also
increments the number of logs logged in the Habits
          * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void BrushingPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).numberOfLogs;

                numberOfLogs2++;

                brushingCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).brushingCount;

                brushingCount2++;

                fullWasherCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).fullWasherCount;

                showerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).showerCount;

                timedShowerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).timedShowerCount;

                offLigtsCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).offLigtsCount;

                matchesCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).matchesCount;

                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    brushingCount = brushingCount2,
                    fullWasherCount = fullWasherCount2,
                    showerCount = showerCount2,
                    timedShowerCount = timedShowerCount2,
                    offLigtsCount = offLigtsCount2,
                    matchesCount = matchesCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
```

```
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2, 
numberOfLogs = 1, brushingCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2, 
numberOfLogs = 1, brushingCount = 1 });
            }
        }
        /** This function updates the points in the Habits category by eight points. It 
also increments the number of logs logged in the Habits
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void DishWasherFullPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).numberOfLogs;

                numberOfLogs2++;

                brushingCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).brushingCount;

                fullWasherCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).fullWasherCount;

                fullWasherCount2++;

                showerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).showerCount;

                timedShowerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).timedShowerCount;

                offLigtsCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).offLigtsCount;

                matchesCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).matchesCount;

                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    brushingCount = brushingCount2,
                    fullWasherCount = fullWasherCount2,
                    showerCount = showerCount2,
                    timedShowerCount = timedShowerCount2,
                    offLigtsCount = offLigtsCount2,
                    matchesCount = matchesCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2,
numberOfLogs = 1, fullWasherCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2, 
numberOfLogs = 1, fullWasherCount = 1 });
            }
        }
        /** This function updates the points in the Habits category by six points. It also 
increments the number of logs logged in the Habits
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void ShowerInsteadPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).numberOfLogs;

                numberOfLogs2++;

                brushingCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).brushingCount;

                fullWasherCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).fullWasherCount;

                showerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).showerCount;

                showerCount2++;

                timedShowerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).timedShowerCount;

                offLigtsCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).offLigtsCount;

                matchesCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).matchesCount;

                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    brushingCount = brushingCount2,
                    fullWasherCount = fullWasherCount2,
                    showerCount = showerCount2,
                    timedShowerCount = timedShowerCount2,
                    offLigtsCount = offLigtsCount2,
                    matchesCount = matchesCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2, 
numberOfLogs = 1, showerCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2, 
numberOfLogs = 1, showerCount = 1 });
            }
        }
        /** This function updates the points in the Habits category by four points. It also 
increments the number of logs logged in the Habits
         * category by one and increments the number of times this particular action was
```

```
logged by one and sends this data to Firebase.
        */
        public async void TimedShowerInsteadPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).numberOfLogs;

                numberOfLogs2++;

                brushingCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).brushingCount;

                fullWasherCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).fullWasherCount;

                showerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).showerCount;

                timedShowerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).timedShowerCount;

                timedShowerCount2++;

                offLigtsCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).offLigtsCount;

                matchesCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).matchesCount;

                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    brushingCount = brushingCount2,
                    fullWasherCount = fullWasherCount2,
                    showerCount = showerCount2,
                    timedShowerCount = timedShowerCount2,
                    offLigtsCount = offLigtsCount2,
                    matchesCount = matchesCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2, numberOfLogs = 1, timedShowerCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2, numberOfLogs = 1, timedShowerCount = 1 });
            }
        }
        /** This function updates the points in the Habits category by two points. It also increments the number of logs logged in the Habits
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void OffLightsPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).numberOfLogs;

                numberOfLogs2++;

                brushingCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).brushingCount;

                fullWasherCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).fullWasherCount;

                showerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).showerCount;

                timedShowerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).timedShowerCount;

                offLigtsCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).offLigtsCount;

                offLigtsCount2++;

                matchesCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).matchesCount;

                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    brushingCount = brushingCount2,
                    fullWasherCount = fullWasherCount2,
                    showerCount = showerCount2,
                    timedShowerCount = timedShowerCount2,
                    offLigtsCount = offLigtsCount2,
                    matchesCount = matchesCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2,
numberOfLogs = 1, offLigtsCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints() { username = username, points = points2,
numberOfLogs = 1, offLigtsCount = 1 });
            }
        }
        /** This function updates the points in the Habits category by two points. It also
increments the number of logs logged in the Habits
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void MatchesPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).numberOfLogs;

                numberOfLogs2++;

                brushingCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).brushingCount;

                fullWasherCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).fullWasherCount;

                showerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).showerCount;

                timedShowerCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).timedShowerCount;

                offLigtsCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).offLigtsCount;

                matchesCount2 = (await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HabitsPoints>()).matchesCount;

                matchesCount2++;

                await firebaseClient
                .Child("HabitsPoints")
                .Child(auth.GetUid())
                .PutAsync(new HabitsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    brushingCount = brushingCount2,
                    fullWasherCount = fullWasherCount2,
                    showerCount = showerCount2,
                    timedShowerCount = timedShowerCount2,
                    offLigtsCount = offLigtsCount2,
                    matchesCount = matchesCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

```

```
                    points2 = AppConstants.twoPoints;
                    await firebaseClient
                    .Child("HabitsPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new HabitsPoints() { username = username, points = points2, 
numberOfLogs = 1, matchesCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.twoPoints;
                    await firebaseClient
                    .Child("HabitsPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new HabitsPoints() { username = username, points = points2, 
numberOfLogs = 1, matchesCount = 1 });
                }
            }
        }
}
```

```
/*! \class The HabitsPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the HomePointsUpdate ViewModel Class. It updates the data for the Home Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class HomePointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int airOutCount2 = 0;
        int nonHarmCount2 = 0;
        int outsideCount2 = 0;
        int plantIntoHomeCount2 = 0;
        int toiletFlushCount2 = 0;

        string username = "";

        IAuth auth;
        /** This function updates the points in the Home category by two points. It also increments the number of logs logged in the Home
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void AirOutPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).numberOfLogs;

                numberOfLogs2++;

                airOutCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).airOutCount;

                airOutCount2++;

                nonHarmCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).nonHarmCount;

                outsideCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).outsideCount;

                plantIntoHomeCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;

                toiletFlushCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).toiletFlushCount;

                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    airOutCount = airOutCount2,
                    nonHarmCount = nonHarmCount2,
                    outsideCount = outsideCount2,
                    plantIntoHomeCount = plantIntoHomeCount2,
                    toiletFlushCount = toiletFlushCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, airOutCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, airOutCount = 1 });
            }
        }
        /** This function updates the points in the Home category by four points. It also
increments the number of logs logged in the Home
        * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void NonHarmfulPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).numberOfLogs;

                numberOfLogs2++;

                airOutCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).airOutCount;

                nonHarmCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).nonHarmCount;

                nonHarmCount2++;

                outsideCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).outsideCount;

                plantIntoHomeCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;

                toiletFlushCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).toiletFlushCount;

                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    airOutCount = airOutCount2,
                    nonHarmCount = nonHarmCount2,
                    outsideCount = outsideCount2,
                    plantIntoHomeCount = plantIntoHomeCount2,
                    toiletFlushCount = toiletFlushCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2, numberOfLogs = 1, nonHarmCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2, numberOfLogs = 1, nonHarmCount = 1 });
            }
        }
        /** This function updates the points in the Home category by two points. It also increments the number of logs logged in the Home
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void OutsidePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).numberOfLogs;

                numberOfLogs2++;

                airOutCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).airOutCount;

                nonHarmCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).nonHarmCount;

                outsideCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).outsideCount;

                outsideCount2++;

                plantIntoHomeCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;

                toiletFlushCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).toiletFlushCount;

                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    airOutCount = airOutCount2,
                    nonHarmCount = nonHarmCount2,
                    outsideCount = outsideCount2,
                    plantIntoHomeCount = plantIntoHomeCount2,
                    toiletFlushCount = toiletFlushCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2, numberOfLogs = 1, outsideCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2, numberOfLogs = 1, outsideCount = 1 });
            }
        }
        /** This function updates the points in the Home category by four points. It also increments the number of logs logged in the Home
        * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void PlantsInsidePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).numberOfLogs;

                numberOfLogs2++;

                airOutCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).airOutCount;

                nonHarmCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).nonHarmCount;

                outsideCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).outsideCount;

                plantIntoHomeCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;

                plantIntoHomeCount2++;

                toiletFlushCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).toiletFlushCount;

                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    airOutCount = airOutCount2,
                    nonHarmCount = nonHarmCount2,
                    outsideCount = outsideCount2,
                    plantIntoHomeCount = plantIntoHomeCount2,
                    toiletFlushCount = toiletFlushCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, plantIntoHomeCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, plantIntoHomeCount = 1 });
            }
        }
        /** This function updates the points in the Home category by four points. It also
increments the number of logs logged in the Home
        * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void ToiletPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).numberOfLogs;

                numberOfLogs2++;

                airOutCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).airOutCount;

                nonHarmCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).nonHarmCount;

                outsideCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).outsideCount;

                plantIntoHomeCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;

                toiletFlushCount2 = (await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<HomePoints>()).toiletFlushCount;

                toiletFlushCount2++;

                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    airOutCount = airOutCount2,
                    nonHarmCount = nonHarmCount2,
                    outsideCount = outsideCount2,
                    plantIntoHomeCount = plantIntoHomeCount2,
                    toiletFlushCount = toiletFlushCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, toiletFlushCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("HomePoints")
                .Child(auth.GetUid())
                .PutAsync(new HomePoints() { username = username, points = points2, numberOfLogs = 1, toiletFlushCount = 1 });
            }
        }
    }
}
```

```
/*! \class The OutdoorsPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the OutdoorsPointsUpdate ViewModel Class. It updates the data for
the Outdoors Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then
sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class OutdoorsPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int campingCount2 = 0;
        int picnicCount2 = 0;
        int plantBushCount2 = 0;
        int plantFlowerCount2 = 0;
        int plantTreeCount2 = 0;
        int scoopCount2 = 0;
        int fruitGardenCount2 = 0;
        int herbGardenCount2 = 0;
        int vegetableGardenCount2 = 0;
        int birdFeederCount2 = 0;

        string username = "";

        IAuth auth;
        /** This function updates the points in the Outdoors category by ten points. It
also increments the number of logs logged in the Outdoors
        * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void CampingPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                campingCount2++;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantBushCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberOfLogs = 1, campingCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberOfLogs = 1, campingCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by six points. It
also increments the number of logs logged in the Outdoors
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
         */
        public async void PicnicPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                picnicCount2++;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantBushCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberOfLogs = 1, picnicCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberOfLogs = 1, picnicCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by eight points. It also increments the number of logs logged in the Outdoors
        * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void PlantBushPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantBushCount2++;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
```

```

                    scoopCount2 = (await firebaseClient
                    .Child("OutdoorsPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                    fruitGardenCount2 = (await firebaseClient
                    .Child("OutdoorsPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                    herbGardenCount2 = (await firebaseClient
                    .Child("OutdoorsPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                    vegetableGardenCount2 = (await firebaseClient
                    .Child("OutdoorsPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                    birdFeederCount2 = (await firebaseClient
                    .Child("OutdoorsPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                    await firebaseClient
                    .Child("OutdoorsPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new OutdoorsPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        campingCount = campingCount2,
                        picnicCount = picnicCount2,
                        plantBushCount = plantBushCount2,
                        plantFlowerCount = plantFlowerCount2,
                        plantTreeCount = plantTreeCount2,
                        scoopCount = scoopCount2,
                        fruitGardenCount = fruitGardenCount2,
                        herbGardenCount = herbGardenCount2,
                        vegetableGardenCount = vegetableGardenCount2,
                        birdFeederCount = birdFeederCount2,

                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("OutdoorsPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberOfLogs = 1, plantBushCount = 1 }); ;

```

```
            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, plantBushCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by ten points. It 
also increments the number of logs logged in the Outdoors
        * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void PlantTreePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                plantTreeCount2++;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantFlowerCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, numberOfLogs = 1, plantTreeCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, numberOfLogs = 1, plantTreeCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by eight points. It also increments the number of logs logged in the Outdoors
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
         */
        public async void PlantFlowerPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantFlowerCount2++;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantFlowerCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, numberOfLogs = 1, plantTreeCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, numberOfLogs = 1, plantTreeCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by four points. It also increments the number of logs logged in the Outdoors
        * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void ScoopPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                scoopCount2++;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantFlowerCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, scoopCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, scoopCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by ten points. It 
also increments the number of logs logged in the Outdoors
        * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void FruitGardenPoints()
        {
```

```csharp

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                fruitGardenCount2++;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantFlowerCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, fruitGardenCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, fruitGardenCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by ten points. It 
also increments the number of logs logged in the Outdoors
        * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void HerbGardenPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                herbGardenCount2++;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantFlowerCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, numberOfLogs = 1, herbGardenCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, numberOfLogs = 1, herbGardenCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by ten points. It also increments the number of logs logged in the Outdoors
        * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void VegetableGardenPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                vegetableGardenCount2++;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantFlowerCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,
 
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, vegetableGardenCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, vegetableGardenCount = 1 });
            }
        }
        /** This function updates the points in the Outdoors category by ten points. It 
also increments the number of logs logged in the Outdoors
        * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void BirdFeederPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;

                numberOfLogs2++;

                campingCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).campingCount;

                picnicCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).picnicCount;

                plantBushCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;

                plantFlowerCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;

                plantTreeCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;

                scoopCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).scoopCount;

                fruitGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;

                herbGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;

                vegetableGardenCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;

                birdFeederCount2 = (await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;

                birdFeederCount2++;

                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    campingCount = campingCount2,
                    picnicCount = picnicCount2,
                    plantBushCount = plantBushCount2,
                    plantFlowerCount = plantFlowerCount2,
                    plantTreeCount = plantTreeCount2,
                    scoopCount = scoopCount2,
                    fruitGardenCount = fruitGardenCount2,
                    herbGardenCount = herbGardenCount2,
                    vegetableGardenCount = vegetableGardenCount2,
                    birdFeederCount = birdFeederCount2,

                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, birdFeederCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("OutdoorsPoints")
                .Child(auth.GetUid())
                .PutAsync(new OutdoorsPoints() { username = username, points = points2, 
numberOfLogs = 1, birdFeederCount = 1 });
            }
        }
    }
}
```

```
/*! \class The PointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the PointsUpdate ViewModel Class. It updates the data for the Overall Points of user for the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class PointsUpdate
    {
        int points2 = 0;
        string username = "";
        string theDate = "";
        long theTime = 0;
        int theCount = 0;
        string currentDate = "";
        long currentTime = 0;


        IAuth auth;
        /** This function increases the points of a user by ten points. It also stores the current date and time under the Security Checks node and also
          * increments the security counter if the point update is on the same day and if it is not then it sets it to zero.
        */
        public async void UpdateByTenPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            currentDate = DateTime.UtcNow.ToString("d");
            currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
            try
            {
                theDate = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).date;

                theTime = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).time;

                theCount = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).counter;

                if (theDate == currentDate)
                {
                    theCount++;
                }
                else
                {
                    theCount = 0;
                }

                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = theCount });

            }
            catch (Exception)
            {
                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime ,
counter = 1});
            }

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .OnceSingleAsync<Points>()).points;

                points2 = points2 + AppConstants.tenPoints;

                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
        }
        /** This function increases the points of a user by eight points. It also stores the current date and time under the Security Checks node and also
         * increments the security counter if the point update is on the same day and if it is not then it sets it to zero.
        */
        public async void UpdateByEightPoints()
        {
            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            currentDate = DateTime.UtcNow.ToString("d");
            currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
            try
            {
                theDate = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).date;

                theTime = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).time;

                theCount = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).counter;

                if (theDate == currentDate)
                {
                    theCount++;
                }
                else
                {
                    theCount = 0;
                }

                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime, counter = theCount });

            }
            catch (Exception)
            {
                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime, 
counter = 1 });
            }

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .OnceSingleAsync<Points>()).points;

                points2 = points2 + AppConstants.eightPoints;

                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
        }
        /** This function increases the points of a user by six points. It also stores the 
current date and time under the Security Checks node and also
         * increments the security counter if the point update is on the same day and if it 
is not then it sets it to zero.
        */
        public async void UpdateBySixPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            currentDate = DateTime.UtcNow.ToString("d");
            currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
            try
            {
                theDate = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).date;

                theTime = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).time;

                theCount = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).counter;

                if (theDate == currentDate)
                {
                    theCount++;
                }
                else
                {
                    theCount = 0;
                }

                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = theCount });

            }
            catch (Exception)
            {
                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = 1 });
            }

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .OnceSingleAsync<Points>()).points;

                points2 = points2 + AppConstants.sixPoints;

                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
        }
        /** This function increases the points of a user by four points. It also stores the current date and time under the Security Checks node and also
         * increments the security counter if the point update is on the same day and if it is not then it sets it to zero.
        */
        public async void UpdateByFourPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            currentDate = DateTime.UtcNow.ToString("d");
            currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
            try
            {
                theDate = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).date;

                theTime = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).time;

                theCount = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).counter;

                if (theDate == currentDate)
                {
                    theCount++;
                }
                else
                {
                    theCount = 0;
                }

                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = theCount });

            }
            catch (Exception)
            {
                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = 1 });
            }

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .OnceSingleAsync<Points>()).points;

                points2 = points2 + AppConstants.fourPoints;

                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
        }
        /** This function increases the points of a user by two points. It also stores the current date and time under the Security Checks node and also
         * increments the security counter if the point update is on the same day and if it is not then it sets it to zero.
        */
        public async void UpdateByTwoPoints()
        {
            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            currentDate = DateTime.UtcNow.ToString("d");
            currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
            try
            {
                theDate = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).date;

                theTime = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).time;

                theCount = (await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .OnceSingleAsync<SecurityChecks>()).counter;

                if (theDate == currentDate)
                {
                    theCount++;
                }
                else
                {
                    theCount = 0;
                }

                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime, counter = theCount });

            }
            catch (Exception)
            {
                await firebaseClient
                .Child("SecurityChecks")
                .Child(auth.GetUid())
                .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = 1 });
            }

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .OnceSingleAsync<Points>()).points;

                points2 = points2 + AppConstants.twoPoints;

                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("Points")
                .Child(auth.GetUid())
                .PutAsync(new Points() { username = username, points = points2 });
            }
        }
    }
}
```

```
/*! \class The SecurityMethods ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SecurityMethods ViewModel Class. It performs security checks for the application whenever a log is made in the application. It
 * prevents the user from logging more than 15 actions per day and more than 1 action per 60 seconds.
 *
 */
using System;
using System.Threading.Tasks;
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class SecurityMethods
    {
        IAuth auth;
        string theDate = "";
        long theTime = 0;
        int theCount = 0;
        string currentDate = "";
        long currentTime = 0;
        private long timeDifference = 0;

        /**
         * This function gets the date and the count from the SecurityChecks Node in the database and compares the stored date to the current date. If
         * the count is 15 and the date is the same as todays date the function returns true. Otherwise False.
         * @return value return true/false
        */
        public async Task<bool> DayLimitLock()
        {
            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            currentDate = DateTime.UtcNow.ToString("d");
            currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();

            try
            {
                theDate = (await firebaseClient
                    .Child("SecurityChecks")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<SecurityChecks>()).date;

                theCount = (await firebaseClient
                    .Child("SecurityChecks")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<SecurityChecks>()).counter;

                //If the count in the database is 15 and the time from the date from the database is the same as today's date return true.
                if (theCount == 15 && theDate == currentDate)
                {
                    return true;
                }

                return false;

            }
            catch (Exception)
            {
                return false;
            }
        }
        /**
         * This function gets the time from the SecurityChecks Node in the database and
compares the stored time to the current time. The time difference
         * is found by subtracting the time stored in the database from the current time
and if the difference is not greater than or equal to 60 seconds then the
         * function returns true otherwise it returns false.
         * @return value return true/false
        */
        public async Task<bool> TimeLimitLock()
        {
            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            currentDate = DateTime.UtcNow.ToString("d");
            currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();

            try
            {
                //Get the time stored in the database.
                theTime = (await firebaseClient
                    .Child("SecurityChecks")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<SecurityChecks>()).time;

                timeDifference = currentTime - theTime;

                if (timeDifference < 60000)
                {
                    return true;
                }

                return false;

            }
            catch (Exception)
            {
                return false;
            }
        }
    }
}
```

```
/*! \class The ShoppingPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ShoppingPointsUpdate ViewModel Class. It updates the data for the Shopping Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class ShoppingPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int clothNapkinCount2 = 0;
        int clothTowelCount2 = 0;
        int applianceCount2 = 0;
        int productCount2 = 0;
        int toothbrushCount2 = 0;
        int clothesCount2 = 0;
        int foodCount2 = 0;
        int localCount2 = 0;
        int looseLeafCount2 = 0;
        int organicFoodCount2 = 0;
        int reusableCount2 = 0;
        int reBatCount2 = 0;
        int reBagCount2 = 0;

        string username = "";

        IAuth auth;
        /** This function updates the points in the Shopping category by two points. It also increments the number of logs logged in the Shopping
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
         */
        public async void ClothNapkinsPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                numberOfLogs2++;

                clothNapkinCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                clothNapkinCount2++;

                clothTowelCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                applianceCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                productCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).productCount;

                toothbrushCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                clothesCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                foodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).foodCount;

                localCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).localCount;

                looseLeafCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                organicFoodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                reusableCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                reBatCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                reBagCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    clothNapkinCount = clothNapkinCount2,
                    clothTowelCount = clothTowelCount2,
                    applianceCount = applianceCount2,
                    productCount = productCount2,
                    toothbrushCount = toothbrushCount2,
                    clothesCount = clothesCount2,
                    foodCount = foodCount2,
                    localCount = localCount2,
                    looseLeafCount = looseLeafCount2,
                    organicFoodCount = organicFoodCount2,
                    reusableCount = reusableCount2,
                    reBatCount = reBatCount2,
                    reBagCount = reBagCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, clothNapkinCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints() { username = username, points = points2, 
numberOfLogs = 1, clothNapkinCount = 1 });
            }
        }
        /** This function updates the points in the Shopping category by two points. It 
also increments the number of logs logged in the Shopping
        * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void ClothTowelsPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                numberOfLogs2++;

                clothNapkinCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                clothTowelCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                clothTowelCount2++;

                applianceCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                productCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).productCount;

                toothbrushCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                clothesCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                foodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).foodCount;

                localCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).localCount;

                looseLeafCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                organicFoodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                reusableCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                reBatCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                reBagCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    clothNapkinCount = clothNapkinCount2,
                    clothTowelCount = clothTowelCount2,
                    applianceCount = applianceCount2,
                    productCount = productCount2,
                    toothbrushCount = toothbrushCount2,
                    clothesCount = clothesCount2,
                    foodCount = foodCount2,
                    localCount = localCount2,
                    looseLeafCount = looseLeafCount2,
                    organicFoodCount = organicFoodCount2,
                    reusableCount = reusableCount2,
                    reBatCount = reBatCount2,
                    reBagCount = reBagCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints() { username = username, points = points2, 
numberOfLogs = 1, clothTowelCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.twoPoints;
                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints() { username = username, points = points2, 
numberOfLogs = 1, clothTowelCount = 1 });
            }
        }
        /** This function updates the points in the Shopping category by two points. It 
also increments the number of logs logged in the Shopping
        * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void AppliancePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).points;

                points2 = points2 + AppConstants.twoPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                numberOfLogs2++;

                clothNapkinCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                clothTowelCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                applianceCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                applianceCount2++;

                productCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).productCount;

                toothbrushCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                clothesCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                foodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).foodCount;

                localCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).localCount;

                looseLeafCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                organicFoodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                reusableCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                reBatCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                reBagCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    clothNapkinCount = clothNapkinCount2,
                    clothTowelCount = clothTowelCount2,
                    applianceCount = applianceCount2,
                    productCount = productCount2,
                    toothbrushCount = toothbrushCount2,
                    clothesCount = clothesCount2,
                    foodCount = foodCount2,
                    localCount = localCount2,
                    looseLeafCount = looseLeafCount2,
                    organicFoodCount = organicFoodCount2,
                    reusableCount = reusableCount2,
                    reBatCount = reBatCount2,
                    reBagCount = reBagCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, applianceCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints() { username = username, points = points2, 
numberOfLogs = 1, applianceCount = 1 });
            }
        }
        /** This function updates the points in the Shopping category by four points. It 
also increments the number of logs logged in the Shopping
        * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void ProductPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                numberOfLogs2++;

                clothNapkinCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                clothTowelCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                applianceCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                productCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).productCount;

                productCount2++;

                toothbrushCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                clothesCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                foodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).foodCount;

                localCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).localCount;

                looseLeafCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                organicFoodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                reusableCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                reBatCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                reBagCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    clothNapkinCount = clothNapkinCount2,
                    clothTowelCount = clothTowelCount2,
                    applianceCount = applianceCount2,
                    productCount = productCount2,
                    toothbrushCount = toothbrushCount2,
                    clothesCount = clothesCount2,
                    foodCount = foodCount2,
                    localCount = localCount2,
                    looseLeafCount = looseLeafCount2,
                    organicFoodCount = organicFoodCount2,
                    reusableCount = reusableCount2,
                    reBatCount = reBatCount2,
                    reBagCount = reBagCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints() { username = username, points = points2, 
numberOfLogs = 1, productCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .PutAsync(new ShoppingPoints() { username = username, points = points2, 
numberOfLogs = 1, productCount = 1 });
            }
        }
        /** This function updates the points in the Shopping category by six points. It 
also increments the number of logs logged in the Shopping
        * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void EcoToothbrushPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                numberOfLogs2++;

                clothNapkinCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                clothTowelCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                applianceCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                productCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).productCount;

                toothbrushCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                toothbrushCount2++;

                clothesCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                foodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).foodCount;

                localCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).localCount;

                looseLeafCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                organicFoodCount2 = (await firebaseClient
                .Child("ShoppingPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
```

```

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.sixPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, 
numberOfLogs = 1, toothbrushCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.sixPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, numberOfLogs = 1, toothbrushCount = 1 });
                }
            }
            /** This function updates the points in the Shopping category by ten points. It also increments the number of logs logged in the Shopping
            * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
            */
            public async void ClothesPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).points;

                    points2 = points2 + AppConstants.tenPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                    numberOfLogs2++;

                    clothNapkinCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                    clothTowelCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                    applianceCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                    productCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).productCount;

                    toothbrushCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                    clothesCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                    clothesCount2++;

                    foodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).foodCount;

                    localCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).localCount;

                    looseLeafCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                    organicFoodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.tenPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
    numberOfLogs = 1, clothesCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.tenPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
    numberOfLogs = 1, clothesCount = 1 });
                }
            }
            /** This function updates the points in the Shopping category by six points. It
    also increments the number of logs logged in the Shopping
            * category by one and increments the number of times this particular action was
    logged by one and sends this data to Firebase.
            */
            public async void FoodInBulkPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
    quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).points;

                    points2 = points2 + AppConstants.sixPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                    numberOfLogs2++;

                    clothNapkinCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                    clothTowelCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                    applianceCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                    productCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).productCount;

                    toothbrushCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                    clothesCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                    foodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).foodCount;

                    foodCount2++;

                    localCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).localCount;

                    looseLeafCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                    organicFoodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.sixPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
    numberOfLogs = 1, foodCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.sixPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, 
    numberOfLogs = 1, foodCount = 1 });
                }
            }
            /** This function updates the points in the Shopping category by eight points. It 
    also increments the number of logs logged in the Shopping
            * category by one and increments the number of times this particular action was 
    logged by one and sends this data to Firebase.
            */
            public async void LocalProductPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
    quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).points;

                    points2 = points2 + AppConstants.eightPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                    numberOfLogs2++;

                    clothNapkinCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                    clothTowelCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                    applianceCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                    productCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).productCount;

                    toothbrushCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                    clothesCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                    foodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).foodCount;

                    localCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).localCount;

                    localCount2++;

                    looseLeafCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                    organicFoodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, numberOfLogs = 1, productCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, numberOfLogs = 1, productCount = 1 });
                }
            }
            /** This function updates the points in the Shopping category by four points. It also increments the number of logs logged in the Shopping
            * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
            */
            public async void TeaPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).points;

                    points2 = points2 + AppConstants.fourPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                    numberOfLogs2++;

                    clothNapkinCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                    clothTowelCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                    applianceCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                    productCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).productCount;

                    toothbrushCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                    clothesCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                    foodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).foodCount;

                    localCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).localCount;

                    looseLeafCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                    looseLeafCount2++;

                    organicFoodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.fourPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
    numberOfLogs = 1, looseLeafCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.fourPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, 
    numberOfLogs = 1, looseLeafCount = 1 });
                }
            }
            /** This function updates the points in the Shopping category by eight points. It 
    also increments the number of logs logged in the Shopping
            * category by one and increments the number of times this particular action was 
    logged by one and sends this data to Firebase.
            */
            public async void OrganicPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
    quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).points;

                    points2 = points2 + AppConstants.eightPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                    numberOfLogs2++;

                    clothNapkinCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                    clothTowelCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                    applianceCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                    productCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).productCount;

                    toothbrushCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                    clothesCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                    foodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).foodCount;

                    localCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).localCount;

                    looseLeafCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                    organicFoodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                    organicFoodCount2++;

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
    numberOfLogs = 1, organicFoodCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
    numberOfLogs = 1, organicFoodCount = 1 });
                }
            }
            /** This function updates the points in the Shopping category by eight points. It
    also increments the number of logs logged in the Shopping
            * category by one and increments the number of times this particular action was
    logged by one and sends this data to Firebase.
            */
            public async void ReWaterPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
    quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).points;

                    points2 = points2 + AppConstants.eightPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                    numberOfLogs2++;

                    clothNapkinCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                    clothTowelCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                    applianceCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                    productCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).productCount;

                    toothbrushCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                    clothesCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                    foodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).foodCount;

                    localCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).localCount;

                    looseLeafCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                    organicFoodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;
```

```

                    reusableCount2++;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, reusableCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, numberOfLogs = 1, reusableCount = 1 });
                }
            }
            /** This function updates the points in the Shopping category by six points. It also increments the number of logs logged in the Shopping
            * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
            */
            public async void ReBattereisPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).points;

                    points2 = points2 + AppConstants.sixPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                    numberOfLogs2++;

                    clothNapkinCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                    clothTowelCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                    applianceCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                    productCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).productCount;

                    toothbrushCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                    clothesCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                    foodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).foodCount;

                    localCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).localCount;

                    looseLeafCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                    organicFoodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBatCount2++;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.sixPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
    numberOfLogs = 1, reBatCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.sixPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2,
    numberOfLogs = 1, reBatCount = 1 });
                }
            }
            /** This function updates the points in the Shopping category by eight points. It
    also increments the number of logs logged in the Shopping
            * category by one and increments the number of times this particular action was
    logged by one and sends this data to Firebase.
            */
            public async void ReBagPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
    quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).points;

                    points2 = points2 + AppConstants.eightPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;

                    numberOfLogs2++;

                    clothNapkinCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;

                    clothTowelCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;

                    applianceCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).applianceCount;

                    productCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).productCount;

                    toothbrushCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;

                    clothesCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).clothesCount;

                    foodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).foodCount;

                    localCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).localCount;

                    looseLeafCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;

                    organicFoodCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;

                    reusableCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reusableCount;

                    reBatCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBatCount;

                    reBagCount2 = (await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<ShoppingPoints>()).reBagCount;

                    reBagCount2++;

                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        clothNapkinCount = clothNapkinCount2,
                        clothTowelCount = clothTowelCount2,
                        applianceCount = applianceCount2,
                        productCount = productCount2,
                        toothbrushCount = toothbrushCount2,
                        clothesCount = clothesCount2,
                        foodCount = foodCount2,
                        localCount = localCount2,
                        looseLeafCount = looseLeafCount2,
                        organicFoodCount = organicFoodCount2,
                        reusableCount = reusableCount2,
                        reBatCount = reBatCount2,
                        reBagCount = reBagCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, 
    numberOfLogs = 1, reBagCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.eightPoints;
                    await firebaseClient
                    .Child("ShoppingPoints")
                    .Child(auth.GetUid())
                    .PutAsync(new ShoppingPoints() { username = username, points = points2, 
    numberOfLogs = 1, reBagCount = 1 });
                }
            }
        }
    }
```

```csharp
/*! \class The TravelPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the TravelPointsUpdate ViewModel Class. It updates the data for the Travel Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class TravelPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int carpoolCount2 = 0;
        int cycleCount2 = 0;
        int ecoCarCount2 = 0;
        int transportCount2 = 0;
        int walkCount2 = 0;


        string username = "";

        IAuth auth;
        /** This function updates the points in the Travel category by six points. It also increments the number of logs logged in the Travel
        * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void CarpoolPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).numberOfLogs;

                numberOfLogs2++;

                carpoolCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).carpoolCount;

                carpoolCount2++;

                cycleCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).cycleCount;

                ecoCarCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).ecoCarCount;

                transportCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).transportCount;

                walkCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).walkCount;

                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    carpoolCount = carpoolCount2,
                    cycleCount = cycleCount2,
                    ecoCarCount = ecoCarCount2,
                    transportCount = transportCount2,
                    walkCount = walkCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, carpoolCount = 1 }); ;

```

```
            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, carpoolCount = 1 });
            }
        }
        /** This function updates the points in the Travel category by ten points. It also
increments the number of logs logged in the Travel
        * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void CyclePoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).numberOfLogs;

                numberOfLogs2++;

                carpoolCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).carpoolCount;

                cycleCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).cycleCount;

                cycleCount2++;

                ecoCarCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).ecoCarCount;

                transportCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).transportCount;

                walkCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).walkCount;

                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    carpoolCount = carpoolCount2,
                    cycleCount = cycleCount2,
                    ecoCarCount = ecoCarCount2,
                    transportCount = transportCount2,
                    walkCount = walkCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, cycleCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, cycleCount = 1 });
            }
        }
        /** This function updates the points in the Travel category by ten points. It also
increments the number of logs logged in the Travel
        * category by one and increments the number of times this particular action was
```

```
logged by one and sends this data to Firebase.
        */
        public async void EcoCarPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).numberOfLogs;

                numberOfLogs2++;

                carpoolCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).carpoolCount;

                cycleCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).cycleCount;

                ecoCarCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).ecoCarCount;

                ecoCarCount2++;

                transportCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).transportCount;

                walkCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).walkCount;

                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    carpoolCount = carpoolCount2,
                    cycleCount = cycleCount2,
                    ecoCarCount = ecoCarCount2,
                    transportCount = transportCount2,
                    walkCount = walkCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, ecoCarCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, ecoCarCount = 1 });
            }
        }
        /** This function updates the points in the Travel category by eight points. It
also increments the number of logs logged in the Travel
        * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void TransportPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).numberOfLogs;

                numberOfLogs2++;

                carpoolCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).carpoolCount;

                cycleCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).cycleCount;

                ecoCarCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).ecoCarCount;

                transportCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).transportCount;

                transportCount2++;

                walkCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).walkCount;

                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    carpoolCount = carpoolCount2,
                    cycleCount = cycleCount2,
                    ecoCarCount = ecoCarCount2,
                    transportCount = transportCount2,
                    walkCount = walkCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, transportCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, transportCount = 1 });
            }
        }
        /** This function updates the points in the Travel category by ten points. It also
increments the number of logs logged in the Travel
        * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void WalkPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).numberOfLogs;

                numberOfLogs2++;

                carpoolCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).carpoolCount;

                cycleCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).cycleCount;

                ecoCarCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).ecoCarCount;

                transportCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).transportCount;

                walkCount2 = (await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<TravelPoints>()).walkCount;

                walkCount2++;

                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    carpoolCount = carpoolCount2,
                    cycleCount = cycleCount2,
                    ecoCarCount = ecoCarCount2,
                    transportCount = transportCount2,
                    walkCount = walkCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2, 
numberOfLogs = 1, walkCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("TravelPoints")
                .Child(auth.GetUid())
                .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, walkCount = 1 });
            }
        }
    }
}
```

```
/*! \class The WastePointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WastePointsUpdate ViewModel Class. It updates the data for the Waste Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class WastePointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int billsCount2 = 0;
        int compostCount2 = 0;
        int setUpRecyclingBinCount2 = 0;
        int bioBinBagsCount2 = 0;
        int recyclingBinCount2 = 0;


        string username = "";

        IAuth auth;
        /** This function updates the points in the Waste category by four points. It also increments the number of logs logged in the Waste
        * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void BillsPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).numberOfLogs;

                numberOfLogs2++;

                billsCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).billsCount;

                billsCount2++;

                compostCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).compostCount;

                setUpRecyclingBinCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;

                bioBinBagsCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).bioBinBagsCount;

                recyclingBinCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).recyclingBinCount;

                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    billsCount = billsCount2,
                    compostCount = compostCount2,
                    setUpRecyclingBinCount = setUpRecyclingBinCount2,
                    bioBinBagsCount = bioBinBagsCount2,
                    recyclingBinCount = recyclingBinCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, billsCount = 1 }); ;

```

```
            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints() { username = username, points = points2, numberOfLogs = 1, billsCount = 1 });
            }
        }
        /** This function updates the points in the Waste category by six points. It also increments the number of logs logged in the Waste
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
         */
        public async void CompostPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).numberOfLogs;

                numberOfLogs2++;

                billsCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).billsCount;

                compostCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).compostCount;

                compostCount2++;

                setUpRecyclingBinCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;

                bioBinBagsCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).bioBinBagsCount;

                recyclingBinCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).recyclingBinCount;

                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    billsCount = billsCount2,
                    compostCount = compostCount2,
                    setUpRecyclingBinCount = setUpRecyclingBinCount2,
                    bioBinBagsCount = bioBinBagsCount2,
                    recyclingBinCount = recyclingBinCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints() { username = username, points = points2, 
numberOfLogs = 1, compostCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints() { username = username, points = points2, 
numberOfLogs = 1, compostCount = 1 });
            }
        }
        /** This function updates the points in the Waste category by ten points. It also 
increments the number of logs logged in the Waste
        * category by one and increments the number of times this particular action was
```

```
logged by one and sends this data to Firebase.
        */
        public async void SetUpRecyclingBinPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).numberOfLogs;

                numberOfLogs2++;

                billsCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).billsCount;

                compostCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).compostCount;

                setUpRecyclingBinCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;

                setUpRecyclingBinCount2++;

                bioBinBagsCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).bioBinBagsCount;

                recyclingBinCount2 = (await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WastePoints>()).recyclingBinCount;

                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    billsCount = billsCount2,
                    compostCount = compostCount2,
                    setUpRecyclingBinCount = setUpRecyclingBinCount2,
                    bioBinBagsCount = bioBinBagsCount2,
                    recyclingBinCount = recyclingBinCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, setUpRecyclingBinCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WastePoints")
                .Child(auth.GetUid())
                .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, setUpRecyclingBinCount = 1 });
            }
        }
        /** This function updates the points in the Waste category by four points. It also
increments the number of logs logged in the Waste
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
         */
        public async void BioBinBagPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WastePoints")
```

```
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).points;

                    points2 = points2 + AppConstants.fourPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).numberOfLogs;

                    numberOfLogs2++;

                    billsCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).billsCount;

                    compostCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).compostCount;

                    setUpRecyclingBinCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;

                    bioBinBagsCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).bioBinBagsCount;

                    bioBinBagsCount2++;

                    recyclingBinCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).recyclingBinCount;

                    await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .PutAsync(new WastePoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        billsCount = billsCount2,
                        compostCount = compostCount2,
                        setUpRecyclingBinCount = setUpRecyclingBinCount2,
                        bioBinBagsCount = bioBinBagsCount2,
                        recyclingBinCount = recyclingBinCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.fourPoints;
                    await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .PutAsync(new WastePoints() { username = username, points = points2, 
    numberOfLogs = 1, bioBinBagsCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.fourPoints;
                    await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .PutAsync(new WastePoints() { username = username, points = points2, 
    numberOfLogs = 1, bioBinBagsCount = 1 });
                }
            }
            /** This function updates the points in the Waste category by six points. It also 
    increments the number of logs logged in the Waste
            * category by one and increments the number of times this particular action was 
    logged by one and sends this data to Firebase.
            */
            public async void RecyclingBinPoints()
            {

                FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
    quake-default-rtdb.firebaseio.com/");
                auth = DependencyService.Get<IAuth>();

                try
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).points;

                    points2 = points2 + AppConstants.sixPoints;

                    numberOfLogs2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).numberOfLogs;

                    numberOfLogs2++;

                    billsCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).billsCount;

                    compostCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).compostCount;

                    setUpRecyclingBinCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;

                    bioBinBagsCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).bioBinBagsCount;

                    recyclingBinCount2 = (await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<WastePoints>()).recyclingBinCount;

                    recyclingBinCount2++;

                    await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .PutAsync(new WastePoints()
                    {
                        username = username,
                        points = points2,
                        numberOfLogs = numberOfLogs2,
                        billsCount = billsCount2,
                        compostCount = compostCount2,
                        setUpRecyclingBinCount = setUpRecyclingBinCount2,
                        bioBinBagsCount = bioBinBagsCount2,
                        recyclingBinCount = recyclingBinCount2,
                    });
                }
                catch (FirebaseException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.sixPoints;
                    await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .PutAsync(new WastePoints() { username = username, points = points2, 
numberOfLogs = 1, recyclingBinCount = 1 }); ;

                }
                catch (NullReferenceException)
                {
                    username = (await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .OnceSingleAsync<Users>()).username;

                    points2 = AppConstants.sixPoints;
                    await firebaseClient
                    .Child("WastePoints")
                    .Child(auth.GetUid())
                    .PutAsync(new WastePoints() { username = username, points = points2,
    numberOfLogs = 1, recyclingBinCount = 1 });
                }
            }
        }
    }
```

```
/*! \class The WaterPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WaterPointsUpdate ViewModel Class. It updates the data for the Water Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class WaterPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int cisternCount2 = 0;
        int rainBarrel2 = 0;
        int reWater2 = 0;
        int showerBucket2 = 0;
        int wSShowerHead2 = 0;

        string username = "";

        IAuth auth;
        /** This function updates the points in the Water category by ten points. It also increments the number of logs logged in the Water
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void CisternPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).numberOfLogs;

                numberOfLogs2++;

                cisternCount2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).cisternCount;

                cisternCount2++;

                rainBarrel2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).rainBarrelCount;

                reWater2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).reWaterCount;

                showerBucket2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).showerBucketCount;

                wSShowerHead2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;

                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    cisternCount = cisternCount2,
                    rainBarrelCount = rainBarrel2,
                    reWaterCount = reWater2,
                    showerBucketCount = showerBucket2,
                    wSShowerHeadCount = wSShowerHead2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, cisternCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, cisternCount = 1 });
            }
        }
        /** This function updates the points in the Water category by ten points. It also
increments the number of logs logged in the Water
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void BarrelPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).numberOfLogs;

                numberOfLogs2++;

                cisternCount2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).cisternCount;

                rainBarrel2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).rainBarrelCount;

                rainBarrel2++;

                reWater2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).reWaterCount;

                showerBucket2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).showerBucketCount;

                wSShowerHead2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;

                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    cisternCount = cisternCount2,
                    rainBarrelCount = rainBarrel2,
                    reWaterCount = reWater2,
                    showerBucketCount = showerBucket2,
                    wSShowerHeadCount = wSShowerHead2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, rainBarrelCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, rainBarrelCount = 1 });
            }
        }
        /** This function updates the points in the Water category by eight points. It also
increments the number of logs logged in the Water
          * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
```

```
        */
        public async void ReWaterPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).numberOfLogs;

                numberOfLogs2++;

                cisternCount2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).cisternCount;

                rainBarrel2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).rainBarrelCount;

                reWater2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).reWaterCount;

                reWater2++;

                showerBucket2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).showerBucketCount;

                wSShowerHead2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;

                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    cisternCount = cisternCount2,
                    rainBarrelCount = rainBarrel2,
                    reWaterCount = reWater2,
                    showerBucketCount = showerBucket2,
                    wSShowerHeadCount = wSShowerHead2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, reWaterCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, reWaterCount = 1 });
            }
        }
        /** This function updates the points in the Water category by eight points. It also
increments the number of logs logged in the Water
         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
        */
        public async void ShowerBucketPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).points;

                points2 = points2 + AppConstants.eightPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).numberOfLogs;

                numberOfLogs2++;

                cisternCount2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).cisternCount;

                rainBarrel2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).rainBarrelCount;

                reWater2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).reWaterCount;

                showerBucket2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).showerBucketCount;

                showerBucket2++;

                wSShowerHead2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;

                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    cisternCount = cisternCount2,
                    rainBarrelCount = rainBarrel2,
                    reWaterCount = reWater2,
                    showerBucketCount = showerBucket2,
                    wSShowerHeadCount = wSShowerHead2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2, numberOfLogs = 1, showerBucketCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.eightPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2, numberOfLogs = 1, showerBucketCount = 1 });
            }
        }
        /** This function updates the points in the Water category by ten points. It also increments the number of logs logged in the Water
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void WSSowerHeadPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).numberOfLogs;

                numberOfLogs2++;

                cisternCount2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).cisternCount;

                rainBarrel2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).rainBarrelCount;

                reWater2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).reWaterCount;

                showerBucket2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).showerBucketCount;

                wSShowerHead2 = (await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;

                wSShowerHead2++;

                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    cisternCount = cisternCount2,
                    rainBarrelCount = rainBarrel2,
                    reWaterCount = reWater2,
                    showerBucketCount = showerBucket2,
                    wSShowerHeadCount = wSShowerHead2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, wSShowerHeadCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WaterPoints")
                .Child(auth.GetUid())
                .PutAsync(new WaterPoints() { username = username, points = points2, 
numberOfLogs = 1, wSShowerHeadCount = 1 });
            }
        }
    }
}
```

```
/*! \class The WorkPointsUpdate ViewModel Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WorkPointsUpdate ViewModel Class. It updates the data for the Work Category of the application. The functions in this class
 * work by reading in all the chosen data and updating the selected fields and then sending this data to back firebase.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using Xamarin.Forms;

namespace Application_Green_Quake.ViewModels
{
    class WorkPointsUpdate
    {
        int points2 = 0;
        int numberOfLogs2 = 0;
        int paperCount2 = 0;
        int offElectronicsCount2 = 0;
        int remoteWorkCount2 = 0;

        string username = "";

        IAuth auth;
        /** This function updates the points in the Work category by four points. It also increments the number of logs logged in the Work
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void PaperPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).points;

                points2 = points2 + AppConstants.fourPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).numberOfLogs;

                numberOfLogs2++;

                paperCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).paperCount;

                paperCount2++;

                offElectronicsCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).offElectronicsCount;

                remoteWorkCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).remoteWorkCount;

                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    paperCount = paperCount2,
                    offElectronicsCount = offElectronicsCount2,
                    remoteWorkCount = remoteWorkCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints() { username = username, points = points2, 
numberOfLogs = 1, paperCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.fourPoints;
                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints() { username = username, points = points2, 
numberOfLogs = 1, paperCount = 1 });
            }
        }
        /** This function updates the points in the Work category by six points. It also increments the number of logs logged in the Work
         * category by one and increments the number of times this particular action was logged by one and sends this data to Firebase.
        */
        public async void ElectonicsOffPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).points;

                points2 = points2 + AppConstants.sixPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).numberOfLogs;

                numberOfLogs2++;

                paperCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).paperCount;

                offElectronicsCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).offElectronicsCount;

                offElectronicsCount2++;

                remoteWorkCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).remoteWorkCount;

                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    paperCount = paperCount2,
                    offElectronicsCount = offElectronicsCount2,
                    remoteWorkCount = remoteWorkCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints() { username = username, points = points2, 
numberOfLogs = 1, offElectronicsCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.sixPoints;
                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints() { username = username, points = points2, 
numberOfLogs = 1, offElectronicsCount = 1 });
            }
        }
        /** This function updates the points in the Work category by ten points. It also 
increments the number of logs logged in the Work
         * category by one and increments the number of times this particular action was 
logged by one and sends this data to Firebase.
        */
        public async void RemoteWorkPoints()
        {

            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
            auth = DependencyService.Get<IAuth>();

            try
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).points;

                points2 = points2 + AppConstants.tenPoints;

                numberOfLogs2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).numberOfLogs;

                numberOfLogs2++;

                paperCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).paperCount;

                offElectronicsCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).offElectronicsCount;

                remoteWorkCount2 = (await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .OnceSingleAsync<WorkPoints>()).remoteWorkCount;

                remoteWorkCount2++;

                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints()
                {
                    username = username,
                    points = points2,
                    numberOfLogs = numberOfLogs2,
                    paperCount = paperCount2,
                    offElectronicsCount = offElectronicsCount2,
                    remoteWorkCount = remoteWorkCount2,
                });
            }
            catch (FirebaseException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints() { username = username, points = points2,
numberOfLogs = 1, remoteWorkCount = 1 }); ;

            }
            catch (NullReferenceException)
            {
                username = (await firebaseClient
                .Child("users")
                .Child(auth.GetUid())
                .OnceSingleAsync<Users>()).username;

                points2 = AppConstants.tenPoints;
                await firebaseClient
                .Child("WorkPoints")
                .Child(auth.GetUid())
                .PutAsync(new WorkPoints() { username = username, points = points2,
numberOfLogs = 1, remoteWorkCount = 1 });
            }
        }
```

```
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             x:Class="Application_Green_Quake.Views.SignUpPage"
             NavigationPage.HasNavigationBar="False">
    <ContentPage.Content>
        <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
            <Grid.RowDefinitions>
                <RowDefinition Height="2*"/>
                <RowDefinition Height="2*"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>

            <Image  Grid.Row="0"
                    Source="{local:ImageResource Application_Green_Quake.Images.Logo.PNG}"
                    Aspect="AspectFit"/>

            <StackLayout Grid.Row="1" BackgroundColor="White" Padding="0,10,0,0">
                <Entry Placeholder="Username"
                       x:Name="UsernameInput"
                       HorizontalTextAlignment="Center"
                       PlaceholderColor="White"
                       TextColor="Black"
                       BackgroundColor="#50C878"
                       Margin="60,0,60,0"/>
                <Label x:Name="UsernameErrorLabel"
                       TextColor="Red"
                       HorizontalTextAlignment="Center"/>
                <Entry Placeholder="Email"
                       Keyboard="Email"
                       x:Name="EmailInput"
                       HorizontalTextAlignment="Center"
                       PlaceholderColor="White"
                       TextColor="Black"
                       BackgroundColor="#50C878"
                       Margin="60,0,60,0"/>
                <Label x:Name="EmailErrorLabel"
                       TextColor="Red"
                       HorizontalTextAlignment="Center"/>
                <Entry Grid.Row="1"
                       Placeholder="Password"
                       HorizontalTextAlignment="Center"
                       IsPassword="true"
                       x:Name="PasswordInput"
                       PlaceholderColor="White"
                       TextColor="Black"
                       BackgroundColor="#50C878"
                       Margin="60,0,60,0"/>
                <Label x:Name="PasswordErrorLabel"
                       Text="Password must be at least 8 chars long and contain an upper
case letter, lower case letter and a number."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"/>
            </StackLayout>
            <StackLayout Grid.Row="2" BackgroundColor="White" VerticalOptions="Center">
                <Button Text="Sign Up"
                        Clicked="SignUpClicked"
                        CornerRadius="30"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"/>
            </StackLayout>
        </Grid>

    </ContentPage.Content>
</ContentPage>
```

```csharp
/*! \class The SignUpPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SignUpPage View Class. This is the class that allows a user to sign up for the application. It contains validation checks.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System;
using System.Globalization;
using System.Text.RegularExpressions;
using Acr.UserDialogs;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SignUpPage : ContentPage
    {
        IAuth auth;
        public SignUpPage()
        {
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();
        }
        /** This function fires when the Sign Up button is clicked. It carries out validation checks and if all are passed the the new user is created. If not
         * the correct error message is displayed.
         */
        async void SignUpClicked(object sender, EventArgs e)
        {
            var emailPattern = "^(?(\")(\".+?(?<!\\\\)\"@)|(([0-9a-z]((\\.(?!\\.))|[-!#\\$%&'\\*\\+/=\\?\\^`\\{\\}\\|~\\w])*)(?<=[0-9a-z])@))(?(\\[)(\\[(\\d{1,3}\\.){3}\\d{1,3}\\])|(([0-9a-z][-\\w]*[0-9a-z]*\\.)+[a-z0-9][\\-a-z0-9]{0,22}[a-z0-9]))$";
            var hasNum = new Regex(@"[0-9]+");
            var hasUpperChar = new Regex(@"[A-Z]+");
            var hasLowerChar = new Regex(@"[a-z]+");
            var hasSpecialChar = new Regex(@"[^\w]+");
            var hasMinimum8Chars = new Regex(@".{8,}");

            UsernameErrorLabel.Text = null;
            EmailErrorLabel.Text = null;
            PasswordErrorLabel.Text = null;

            if (UsernameInput.Text == null && EmailInput.Text != null && PasswordInput.Text != null)
            {

                UsernameInput.Text = null;
                UsernameErrorLabel.Text = "No Username Entered";
            }
            else if (UsernameInput.Text != null && EmailInput.Text == null && PasswordInput.Text != null)
            {
                EmailInput.Text = null;
                EmailErrorLabel.Text = "No Email Entered";
            }
            else if (UsernameInput.Text != null && EmailInput.Text != null && PasswordInput.Text == null)
            {

                PasswordInput.Text = null;
                PasswordErrorLabel.TextColor = Color.Red;
                PasswordErrorLabel.Text = "No Password Entered";
            }
            else if (UsernameInput.Text != null && EmailInput.Text == null && PasswordInput.Text == null)
            {
                EmailInput.Text = null;
                PasswordInput.Text = null;
                PasswordErrorLabel.TextColor = Color.Red;
                PasswordErrorLabel.Text = "No Password Entered";
                EmailErrorLabel.Text = "No Email Entered";
            }
            else if (UsernameInput.Text == null && EmailInput.Text != null && PasswordInput.Text == null)
            {
                PasswordInput.Text = null;
                UsernameInput.Text = null;
                PasswordErrorLabel.TextColor = Color.Red;
                UsernameErrorLabel.Text = "No Username Entered";
                PasswordErrorLabel.Text = "No Password Entered";
                    await DisplayAlert("Sign Up Failed", "No Username or  Password", "Ok");
            }
            else if (UsernameInput.Text == null && EmailInput.Text == null && PasswordInput.Text != null)
            {
                EmailInput.Text = null;

                UsernameInput.Text = null;
                UsernameErrorLabel.Text = "No Username Entered";
                EmailErrorLabel.Text = "No Email Entered";
            }
            else if (UsernameInput.Text == null && EmailInput.Text == null && PasswordInput.Text == null)
            {
                EmailInput.Text = null;
                PasswordInput.Text = null;
                UsernameInput.Text = null;
                PasswordErrorLabel.TextColor = Color.Red;
                EmailErrorLabel.Text = "No Email Entered";
                PasswordErrorLabel.Text = "No Password Entered";
                UsernameErrorLabel.Text = "No Username Entered";
            }

            if (UsernameInput.Text != null && EmailInput.Text != null && PasswordInput.Text != null)
            {
                if (!Regex.IsMatch(EmailInput.Text, emailPattern))
                {
                    EmailInput.Text = null;
                    EmailErrorLabel.Text = "Email is invalid";
                }
                if (!hasNum.IsMatch(PasswordInput.Text))
                {
                    PasswordInput.Text = null;
                    PasswordErrorLabel.TextColor = Color.Red;
                    PasswordErrorLabel.Text = "Password must have at least one number";
                }
                else if (!hasLowerChar.IsMatch(PasswordInput.Text))
                {
                    PasswordInput.Text = null;
                    PasswordErrorLabel.TextColor = Color.Red;
                    PasswordErrorLabel.Text = "Password must have at least one lower case character";
                }
                else if (!hasUpperChar.IsMatch(PasswordInput.Text))
                {
                    PasswordInput.Text = null;
                    PasswordErrorLabel.TextColor = Color.Red;
                    PasswordErrorLabel.Text = "Password must have at least one upper case character";
                }
                else if (!hasSpecialChar.IsMatch(PasswordInput.Text))
                {
                    PasswordInput.Text = null;
                    PasswordErrorLabel.TextColor = Color.Red;
                    PasswordErrorLabel.Text = "Password must have at least one special character";
                }
                else if (!hasMinimum8Chars.IsMatch(PasswordInput.Text))
                {
                    PasswordInput.Text = null;
                    PasswordErrorLabel.TextColor = Color.Red;
                    PasswordErrorLabel.Text = "Password must be at least 8 characters";
                }
                else if (EmailErrorLabel.Text == null && PasswordErrorLabel.Text == null && UsernameErrorLabel.Text == null)
                {

                    var user = auth.SignUpWithEmailAndPassword(EmailInput.Text, PasswordInput.Text);
                    if (user != null)
                    {
                        var signOut = auth.SignOut();

                        FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");

                        string usernameInput = UsernameInput.Text;
                        string token = await user;
                        string theBio = "";
                        string theNation = RegionInfo.CurrentRegion.EnglishName;

                        if (token != "duplicate")
                        {
                            UserDialogs.Instance.ShowLoading("");

                            await firebaseClient
                                .Child("users")
                                .Child(token)
                                .PutAsync(new Users() { username = usernameInput, bio = theBio, nation = theNation});

                            await firebaseClient
                                .Child("usernames")
                                .Child(usernameInput)
                                .PutAsync(new Usernames() { Uid = token });

                            if (signOut)
                            {
                                UserDialogs.Instance.HideLoading();
                                await DisplayAlert("Success", "New User Created", "OK");
                                await Navigation.PushAsync(new MainPage());
                            }
                            else
                            {
                                await DisplayAlert("Error", "An error has occurred, please try again", "Ok");
                            }
                        }
                        else
                        {
                            await DisplayAlert("Error", "The email already exists, please try again.", "Ok");
                        }
                    }
                    else
                    {
                        await DisplayAlert("Error", "Please connect to the internet.", "Ok");
                    }
                }
            }
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             x:Class="Application_Green_Quake.Views.MainPage"
             NavigationPage.HasNavigationBar="False">
    <ContentPage.Content>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image
                    Grid.Row="0"
                    Source="{local:ImageResource Application_Green_Quake.Images.Logo.PNG}"
                    Aspect="AspectFit"/>

                <StackLayout Grid.Row="1" BackgroundColor="White">
                    <Entry Placeholder="Email"
                           Keyboard="Email"
                           x:Name="EmailInput"
                           HorizontalTextAlignment="Center"
                           PlaceholderColor="White"
                           TextColor="Black"
                           BackgroundColor="#50C878"
                           Margin="60,0,60,0"/>
                    <Label x:Name="EmailErrorLabel"
                           TextColor="Red"
                           HorizontalTextAlignment="Center"/>
                    <Entry Grid.Row="1"
                           Placeholder="Password"
                           HorizontalTextAlignment="Center"
                           IsPassword="true"
                           x:Name="PasswordInput"
                           PlaceholderColor="White"
                           TextColor="Black"
                           BackgroundColor="#50C878"
                           Margin="60,0,60,0"/>
                    <Label x:Name="PasswordErrorLabel"
                           TextColor="Red"
                           HorizontalTextAlignment="Center"/>
            </StackLayout>
                <StackLayout Grid.Row="2" BackgroundColor="White">
                    <Button Text="Login"
                            Clicked="LoginClicked"
                            CornerRadius="30"
                            BackgroundColor="#50C878"
                            TextColor="White"
                            Margin="60,0,60,0"/>
                    <Button Text="Sign Up"
                            TextColor="#50C878"
                            HorizontalOptions="Center"
                            BackgroundColor="Transparent"
                            Clicked="SignUpClicked" />
                    <Button Text="Forgot Password?"
                            HorizontalOptions="Center"
                            BackgroundColor="Transparent"
                            Clicked="ForgotPasswordClicked"
                            Padding="0,0,0,30"
                            TextColor="Black"/>
                </StackLayout>
            </Grid>
    </ContentPage.Content>
</ContentPage>
```

```csharp
/*! \class The MainPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the MainPage View Class. This is the class that allows a user to login to the application. It contains validation checks.
 *
 */
using System;
using Application_Green_Quake.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainPage : ContentPage
    {
        IAuth auth;
        public static string token;
        public MainPage()
        {
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();
        }
        /** This function fires when the Login button is clicked. It carries out validation checks and if all are passed the the user is logged in. If not
          * the correct error message is displayed.
        */
        async void LoginClicked(object sender, EventArgs e)
        {
            EmailErrorLabel.Text = null;
            PasswordErrorLabel.Text = null;

            if (EmailInput.Text == null && PasswordInput.Text != null)
            {
                EmailInput.Text = null;
                PasswordInput.Text = null;
                EmailErrorLabel.Text = "No Email Entered";
            }
            else if (PasswordInput.Text == null && EmailInput.Text != null)
            {
                EmailInput.Text = null;
                PasswordInput.Text = null;
                PasswordErrorLabel.Text = "No Password Entered";
            }
            else if (EmailInput.Text == null && PasswordInput.Text == null)
            {
                EmailInput.Text = null;
                PasswordInput.Text = null;
                EmailErrorLabel.Text = "No Email Entered";
                PasswordErrorLabel.Text = "No Password Entered";
            }
            else
            {
                try
                {
                    token = await auth.LoginWithEmailAndPassword(EmailInput.Text, PasswordInput.Text);
                    if (token != string.Empty)
                    {
                        GetData level = new GetData();
                        level.SetLvl();
                        await Navigation.PushAsync(new LoginSplashPage());
                    }
                    else
                    {
                        EmailInput.Text = null;
                        PasswordInput.Text = null;
                        await DisplayAlert("Authentication Failed", "Email or Password are incorrect", "Ok");
                    }
                }
                catch (Exception)
                {
                    await DisplayAlert("Authentication Failed", "Please connect to the internet", "Ok");
                }

            }
        }
        /** This function fires when the Sign Up Text is clicked. It signs out the user if a user is signed in and redirects to the Sign Up page.
        */
        void SignUpClicked(object sender, EventArgs e)
        {
            var signOut = auth.SignOut();

            if (signOut)
            {
                Navigation.PushAsync(new SignUpPage());
            }
        }
        /** This function fires when the Forgot Password Text is clicked. It redirects to the Forgot Password page.
        */
        private async void ForgotPasswordClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ForgotPasswordPage());
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>

<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:views="clr-
namespace:Application_Green_Quake.Views"
            xmlns:local="clr-namespace:Application_Green_Quake.Models" xmlns:views1="clr-
namespace:Application_Green_Quake.Views.ProfilePage" xmlns:views2="clr-
namespace:Application_Green_Quake.Views.LeaderboardPage"
            x:Class="Application_Green_Quake.Views.MainMenu"
            NavigationPage.HasBackButton="False"
            xmlns:android="clr-
namespace:Xamarin.Forms.PlatformConfiguration.AndroidSpecific;assembly=Xamarin.Forms.Core"
            android:TabbedPage.ToolbarPlacement="Bottom"
            UnselectedTabColor="Black"
            SelectedTabColor="White" >

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Green Quake" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage Title="Home" IconImageSource="{local:ImageResource
Application_Green_Quake.Images.Tabbed.Home.png}">
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Grid.Column="0"
                   Grid.Row="0"
                   Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.EcoActions.jpg}"/>
                <StackLayout Grid.Column="0"
                         Grid.Row="0"
                         BackgroundColor="Black"
                         Opacity=".4">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoActionButton"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                         Grid.Row="0"
                         VerticalOptions="Center"
                         Spacing="5"
                         Margin="15,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoActionButton"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Eco Actions"
                       TextColor="White"
                       FontSize="24"
```

```
                      FontAttributes="Bold"
                      FontFamily="Proxima Nova"/>
                   <Label Text="Log your Eco Friendly Activities and earn rewards!"
                      TextColor="White"
                      FontSize="15"
                      FontFamily="Proxima Nova Thin"/>
               </StackLayout>

               <Image Grid.Column="0"
                  Grid.Row="1"
                  Aspect="AspectFill"
                  Source="{local:ImageResource
Application_Green_Quake.Images.Refill.jpg}"/>
               <StackLayout Grid.Column="0"
                        Grid.Row="1"
                        BackgroundColor="Black"
                        Opacity=".4">
                   <StackLayout.GestureRecognizers>
                       <TapGestureRecognizer Tapped="NavigateToRefillStation"/>
                   </StackLayout.GestureRecognizers>
               </StackLayout>
               <StackLayout Grid.Column="0"
                        Grid.Row="1"
                        VerticalOptions="Center"
                        Spacing="5"
                        Margin="15,0,40,15">
                   <StackLayout.GestureRecognizers>
                       <TapGestureRecognizer Tapped="NavigateToRefillStation"/>
                   </StackLayout.GestureRecognizers>
                   <Label Text="Refill Stations"
                      TextColor="White"
                      FontSize="24"
                      FontAttributes="Bold"
                      FontFamily="Proxima Nova"/>
                   <Label Text="Find the closest place where you can get a refill. Use
reusable bottles and reduce plastic waste!"
                      TextColor="White"
                      FontSize="15"
                      FontFamily="Proxima Nova Thin" />
               </StackLayout>

               <Image Grid.Column="0"
                  Grid.Row="2"
                  Aspect="AspectFill"
                  Source="{local:ImageResource
Application_Green_Quake.Images.SignOut.jpg}"/>
               <StackLayout Grid.Column="0"
                        Grid.Row="2"
                        BackgroundColor="Black"
                        Opacity=".4">
                   <StackLayout.GestureRecognizers>
                       <TapGestureRecognizer Tapped="SignOutButton"/>
                   </StackLayout.GestureRecognizers>
               </StackLayout>
               <StackLayout Grid.Column="0"
                        Grid.Row="2"
                        VerticalOptions="Center"
                        Spacing="5"
                        Margin="15,0,40,15">
                   <StackLayout.GestureRecognizers>
                       <TapGestureRecognizer Tapped="SignOutButton"/>
                   </StackLayout.GestureRecognizers>
```

```xml
                        <Label Text="Sign Out"
                           TextColor="White"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    </StackLayout>
                </Grid>
            </ScrollView>
        </ContentPage>

        <views2:TopTabLeaderBoard Title="Leaderboard" IconImageSource="{local:ImageResource Application_Green_Quake.Images.Tabbed.Leaderboard.png}"></views2:TopTabLeaderBoard>
        <views1:TopTabProfile Title="Profile" IconImageSource="{local:ImageResource Application_Green_Quake.Images.Tabbed.Profile.png}"></views1:TopTabProfile>
</TabbedPage>
```

```
/*! \class The MainMenu View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the MainMenu View Class. This page is the main menu of the application and provides navigation to all the apps screens.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.EcoActionsMenu;
using Application_Green_Quake.Views.RefillPage;
using System;
using System.Threading.Tasks;
using Acr.UserDialogs;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using Application = Xamarin.Forms.Application;

namespace Application_Green_Quake.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainMenu : TabbedPage
    {
        IAuth auth;
        public MainMenu()
        {
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();
            OnAppearing();
        }

        /** The constructor for Main menu
        @param tab supplied to tell the class which tabbed page to display.
        */
        public MainMenu(int tab)
        {
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();
            CurrentPage = Children[tab];
            OnAppearing();
        }

        /** This function navigates to ActionCategories
        */
        private async void NavigateToEcoActionButton(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ActionsCategories());
        }
        /** This function navigates to RefillStation
        */
        private async void NavigateToRefillStation(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new RefillStation());
        }

        /** This function Signs the user out and navigates to MainPage
        */
        void SignOutButton(object sender, EventArgs e)
        {
            var signOut = auth.SignOut();

            if (signOut)
            {
                Application.Current.MainPage  = new NavigationPage(new MainPage());
            }
        }
        /** This function is called before the page is displayed.
        */
        protected override async void OnAppearing()
        {
            // Wait 2 seconds to allow the data to load.
            await Task.Delay(2000);
            //Call functions to load and set data.
            GetData data = new GetData();
            data.SetLvl();
            UserDialogs.Instance.HideLoading();

            GetBadgeData badgeData = new GetBadgeData();
            badgeData.SetBadgeData();

            GetAchievementsData achievementsData = new GetAchievementsData();
            achievementsData.SetAchievementsData();

            //Set the level in the navigation bar.
            theLevel.Text = "LVL: " + GetData.lvl;

        }
    }
}
```

```csharp
/*! \class The LoginSplashPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the LoginSplashPage View Class. This class provides the splash screen when a user logs into the applciation.
 *
 */
using Xamarin.Forms;

namespace Application_Green_Quake.Views
{
    public class LoginSplashPage : ContentPage
    {
        Image splashImage;

        /** This function fires when the Login button is clicked. It provides the splash screen and then navigates to the main menu right after.
        */
        public LoginSplashPage()
        {
            NavigationPage.SetHasNavigationBar(this, false);

            //Set the image for the splash screen
            var sub = new AbsoluteLayout();
            splashImage = new Image
            {
                Source = ImageSource.FromResource("Application_Green_Quake.Images.trees.png"),
                WidthRequest =  150,
                HeightRequest =  150
            };

            AbsoluteLayout.SetLayoutFlags(splashImage,
                AbsoluteLayoutFlags.PositionProportional);
            AbsoluteLayout.SetLayoutBounds(splashImage,
                new Rectangle(0.5, 0.5, AbsoluteLayout.AutoSize,AbsoluteLayout.AutoSize));

            sub.Children.Add(splashImage);

            this.BackgroundColor = Color.FromHex("#50C878");
            this.Content = sub;
        }

        protected override async void OnAppearing()
        {
            base.OnAppearing();
            //Set the animation
            await splashImage.ScaleTo(0.4, 1100, Easing.Linear);
            await splashImage.ScaleTo(700, 900, Easing.Linear);
            Application.Current.MainPage = new NavigationPage(new MainMenu());
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.ForgotPasswordPage"
             NavigationPage.HasNavigationBar="False">
    <ContentPage.Content>
        <StackLayout VerticalOptions="CenterAndExpand" Spacing="20">
            <Label Text="Forgot Your Password"
                   HorizontalOptions="CenterAndExpand"
                   TextColor="Black"
                   FontSize="25"
                   FontAttributes="Bold"/>
            <Entry Placeholder="Email"
                   Keyboard="Email"
                   x:Name="EmailInput"
                   HorizontalTextAlignment="Center"
                   PlaceholderColor="White"
                   TextColor="Black"
                   BackgroundColor="#50C878"
                   Margin="60,0,60,0"/>
            <Label x:Name="EmailErrorLabel"
                   TextColor="Red"
                   HorizontalTextAlignment="Center"/>
            <Button Text="Send"
                    Clicked="OnResetPassword"
                    CornerRadius="30"
                    BackgroundColor="#50C878"
                    TextColor="White"
                    Margin="60,0,60,0"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.ForgotPasswordPage"
             NavigationPage.HasNavigationBar="False">
    <ContentPage.Content>
        <StackLayout VerticalOptions="CenterAndExpand" Spacing="20">
            <Label Text="Forgot Your Password"
                   HorizontalOptions="CenterAndExpand"
                   TextColor="Black"
                   FontSize="25"
                   FontAttributes="Bold"/>
            <Entry Placeholder="Email"
                   Keyboard="Email"
                   x:Name="EmailInput"
                   HorizontalTextAlignment="Center"
                   PlaceholderColor="White"
                   TextColor="Black"
                   BackgroundColor="#50C878"
                   Margin="60,0,60,0"/>
            <Label x:Name="EmailErrorLabel"
                   TextColor="Red"
                   HorizontalTextAlignment="Center"/>
            <Button Text="Send"
                    Clicked="OnResetPassword"
                    CornerRadius="30"
                    BackgroundColor="#50C878"
                    TextColor="White"
                    Margin="60,0,60,0"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:maps="clr-
namespace:Xamarin.Forms.GoogleMaps;assembly=Xamarin.Forms.GoogleMaps"
             x:Class="Application_Green_Quake.Views.RefillPage.RefillStation"
             NavigationPage.HasNavigationBar="False">
    <ContentPage.Content>
            <maps:Map x:Name ="map"/>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The RefillStation View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the RefillStation View Class. This page contains the map and loads data from firebase and then uses this data to display the pins.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using System;
using System.Diagnostics;
using System.Linq;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.GoogleMaps;
using Xamarin.Forms.Xaml;
namespace Application_Green_Quake.Views.RefillPage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RefillStation : ContentPage
    {
        public RefillStation()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function is called before the page is displayed.
        */
        protected override async void OnAppearing()
        {
            try
            {
                var location = await Geolocation.GetLastKnownLocationAsync();
                if (location == null)
                {
                    location = await Geolocation.GetLocationAsync(new GeolocationRequest
                    {
                        DesiredAccuracy = GeolocationAccuracy.Medium,
                        Timeout = TimeSpan.FromSeconds(30)
                    });


                }
                else
                {
                    Pin currentLocation = new Pin()
                    {
                        Type = PinType.SavedPin,
                        Label = "Me",
                        Address = "Here",
                        Position = new Position(location.Latitude, location.Longitude),
                        Tag = "id_Me",

                    };
                    // Add the pin and load the map at this location
                    map.Pins.Add(currentLocation);
                    map.MoveToRegion(MapSpan.FromCenterAndRadius(currentLocation.Position, Distance.FromMeters(5000)));

                }

            }
            catch (Exception e)
            {
                Debug.WriteLine($"Something is wrong: {e.Message}");
            }


            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");

            //Load the pin data into a list
            var list = (await firebaseClient
           .Child("Stations")
           .OnceAsync<Station>()).Select(item => new Station
           {
               description = item.Object.description,
               label = item.Object.label,
               latitude = item.Object.latitude,
               longitude = item.Object.longitude,

           }).ToList();

            // For each entry in the data create and place a pin.
            foreach (var obj in list)
            {
                Pin stationLocations = new Pin()
                {
                    Type = PinType.SavedPin,
                    Label = obj.label,
                    Address = obj.description,
                    Position = new Position(obj.latitude, obj.longitude),

                };
                map.Pins.Add(stationLocations);
            }
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Home.AirOutHome"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Air Out Home" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.AirOut.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Air out your home and get rid of those indoor pollutants."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="2 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The AirOutHome View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the AirOutHome View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Home
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AirOutHome : ContentPage
    {
        public AirOutHome()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTwoPoints();
                HomePointsUpdate helper2 = new HomePointsUpdate();
                helper2.AirOutPoints();
                await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Waste.BillsOnline"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Online Bills" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.OnlineBills.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Pay your bills online. There is no need to get letters 
anymore. This removes the need to produce more paper and print letters an in turn reduces 
paper usage and waste."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The BillsOnline View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the BillsOnline View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Waste
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class BillsOnline : ContentPage
    {
        public BillsOnline()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                WastePointsUpdate helper2 = new WastePointsUpdate();
                helper2.BillsPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Work.BothSidesPaper"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Both Sides" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Work.Paper.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Use both sides of the paper. This reduces paper waste and the amount of paper needed."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Work
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class BothSidesPaper : ContentPage
    {
        public BothSidesPaper()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                WorkPointsUpdate helper2 = new WorkPointsUpdate();
                helper2.PaperPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Habits.BrushingTeeth"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Tap Off" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.TimedBrushing.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="When brushing your teeth turn of the water. This helps save 
water while water waste is becoming a major problem in todays world. Water shortages and 
draught are increasing throughout the globe."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="2 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The BrushingTeeth View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the BrushingTeeth View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Habits
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class BrushingTeeth : ContentPage
    {
        public BrushingTeeth()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTwoPoints();
                HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
                helper2.BrushingPoints();
                await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.BuyOrganicFood"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Go Organic" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.OrganicFood.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and consume organic food. It contains less 
chemicals and is healthier than non organic food."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The BuyOrganicFood View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the BuyOrganicFood View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class BuyOrganicFood : ContentPage
    {
        public BuyOrganicFood()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.OrganicPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Travel.Carpool"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Carpool" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.Carpool.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Carpool instead of going alone. This reduces emissions and 
can be quiet fun too."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The Carpool View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Carpool View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Travel
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Carpool : ContentPage
    {
        public Carpool()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                TravelPointsUpdate helper2 = new TravelPointsUpdate();
                helper2.CarpoolPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Water.CisternDisplacement"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Cistern System" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Water.CisternDis.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Install a Cistern Displacement Device to use less water and 
help reduce water waste."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The CisternDisplacement View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the CisternDisplacement View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Water
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CisternDisplacement : ContentPage
    {
        public CisternDisplacement()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                WaterPointsUpdate helper2 = new WaterPointsUpdate();
                helper2.CisternPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.ClothNapkins"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Cloth Napkins" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.Napkin.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and use cloth napkins instead of paper ones. Paper 
napkins contribute to waste and have to be remade. Cloth napkins can be washed and reused 
and are biodegradable."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="2 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ClothNapkins View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ClothNapkins View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ClothNapkins : ContentPage
    {
        public ClothNapkins()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTwoPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.ClothNapkinsPoints();
                await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.ClothTowels"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Cloth Towels" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.Towel.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and use cloth towels instead of paper ones. Paper 
towels contribute to waste and have to be remade. Cloth towels can be washed and reused and 
are biodegradable."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="2 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ClothTowels View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ClothTowels View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ClothTowels : ContentPage
    {
        public ClothTowels()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTwoPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.ClothTowelsPoints();
                await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Waste.CompostWaste"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Compost Bins" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.Compost.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Compost you food waste. Adding compost made from organic 
materials to your garden improves moisture retention, provides slow-release nutrients and 
reduce pesticide problem. Healthier plants grown from healthier soil are a huge and visible 
bonus for all the keen gardeners out there."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The CompostWaste View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the CompostWaste View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Waste
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CompostWaste : ContentPage
    {
        public CompostWaste()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                WastePointsUpdate helper2 = new WastePointsUpdate();
                helper2.CompostPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="Application_Green_Quake.Views.EcoActions.Community.CreateEnvironmentalGroup"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Create Group" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Community.CreateE.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="No Environmental Group to join? Just create one and meet 
and gather like minded people!"
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The CreateEnvironmentalGroup View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the CreateEnvironmentalGroup View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Models;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Threading.Tasks;

namespace Application_Green_Quake.Views.EcoActions.Community
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CreateEnvironmentalGroup : ContentPage
    {
        public CreateEnvironmentalGroup()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
                helper2.CreateGroupPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Travel.Cycle"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Cycle" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Cycle.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Cycle there if you can. Cycling is great for your overall
fitness and has no negative impacts on the environment."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The Cycle View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Cycle View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Travel
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Cycle : ContentPage
    {
        public Cycle()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                TravelPointsUpdate helper2 = new TravelPointsUpdate();
                helper2.CyclePoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Habits.DishwasherFull"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Full Dishwasher" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.FullDishwasher.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Only use the dish washer when it is full. Believe it or not
dishwasher are more efficient than hand washing but only when they are loaded as they end up
using less water, less energy and more money."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The DishwasherFull View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the DishwasherFull View Class. This class is the eco action that the 
user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Habits
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class DishwasherFull : ContentPage
    {
        public DishwasherFull()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
                helper2.DishWasherFullPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Community.DoCommunity"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Community Work" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Community.DoCommunity.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Do something for the community and spread positivity. Small 
things matter."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The DoCommunity View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the DoCommunity View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Community
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class DoCommunity : ContentPage
    {
        public DoCommunity()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
                helper2.CommunityPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Community.DonateItems"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Donate" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Community.Donate.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Instead of throwing items away or hoarding them it is 
definitely a better option to give them away to the less fortunate. Not only will you be 
doing a good deed but you will also feel great."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The DonateItems View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the DonateItems View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Community
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class DonateItems : ContentPage
    {
        public DonateItems()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
                helper2.DonatePoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.DryerFull"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Full Dryer" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.FullDryer.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Only use the dryer when it is full with clothes to save 
energy. It is estimated that a dryer emits more than a ton of CO2 per year and contributes 
to major energy waste. If possible skip using the dryer altogether."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The DryerFull View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the DryerFull View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class DryerFull : ContentPage
    {
        public DryerFull()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.DryerFullPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.EatAllYouMake"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Eat All" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.EatAllYouMake.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Only make enough food so you can eat it all. This reduces 
food waste which is a major concern.  When food rots it emits methane, a powerful greenhouse 
gas which is 25 times worse for the ozone layer than CO2. 60% of methane is produced by 
humans."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EatAllYouMake View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EatAllYouMake View Class. This class is the eco action that the
user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EatAllYouMake : ContentPage
    {
        public EatAllYouMake()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.EatAllPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EcoFreidnlyApplicance"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Eco Appliance" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.EcoAppliance.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and use an Eco Friendly appliance. This uses less
energy and is better for the environment."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EcoFreidnlyApplicance View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EcoFreidnlyApplicance View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EcoFreidnlyApplicance : ContentPage
    {
        public EcoFreidnlyApplicance()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.AppliancePoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Travel.EcoFreindlyCar"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Eco Car" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.EcoCar.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and use an Eco Friendly car. These types of cars 
are more Environmentally Friendly that other cars. Just by simply owning one you are doing 
more for the environment than people who do not own one."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EcoFreindlyCar View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EcoFreindlyCar View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Travel
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EcoFreindlyCar : ContentPage
    {
        public EcoFreindlyCar()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                TravelPointsUpdate helper2 = new TravelPointsUpdate();
                helper2.EcoCarPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EcoFriendlyProduct"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Eco Product" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Shopping.EcoProduct.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and use an Eco Friendly product that has no bad chemicals and is manufactured in an eco friendlily way. This helps the environment and means you are doing your part."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EcoFriendlyProduct View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EcoFriendlyProduct View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EcoFriendlyProduct : ContentPage
    {
        public EcoFriendlyProduct()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.ProductPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EcoFriendlyToothbrush"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Eco Toothbrush" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.EcoBrush.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and use an Eco Friendly toothbrush and toothpaste.
This is something we do at least twice a day so it is a good idea to do it in a Eco Friendly
Fashion."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EcoFriendlyToothbrush View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EcoFriendlyToothbrush View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EcoFriendlyToothbrush : ContentPage
    {
        public EcoFriendlyToothbrush()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.EcoToothbrushPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.EfficientThermostat"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Efficient Thermostat" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.Thermostat.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Efficiently program your thermostat so it saves energy. Do 
you ever have the heating on when you are not at home or the rooms are too warm?"
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EfficientThermostat View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EfficientThermostat View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EfficientThermostat : ContentPage
    {
        public EfficientThermostat()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.EfficientThermostatPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             
x:Class="Application_Green_Quake.Views.EcoActions.Community.EnvironmentalGroups"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Join Group" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Community.JoinE.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Join an environmental group to discuss or to take action 
for the environment."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EnvironmentalGroups View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EnvironmentalGroups View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Community
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EnvironmentalGroups : ContentPage
    {
        public EnvironmentalGroups()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
                helper2.GroupPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EthicalClothes"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Ethical Clothes" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.EthicalClothes.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and wear ethical clothes. Ethical Clothing is an 
umbrella term to describe ethical fashion design, production, retail, and purchasing. It 
covers a range of issues such as working conditions, exploitation, fair trade, sustainable 
production, the environment, and animal welfare."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EthicalClothes View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EthicalClothes View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EthicalClothes : ContentPage
    {
        public EthicalClothes()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.ClothesPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
x:Class="Application_Green_Quake.Views.EcoActions.AdvancedPageItems.FixInsteadOfThrowAway">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Fix It" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Advanced.Fix.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Instead of throwing an item away try and fix it. This is
much more environmentally friendly. If it cannot be fixed try and turn it into something
else"
                   TextColor="Black"
                   HorizontalTextAlignment="Center"
                   FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The FixInsteadOfThrowAway View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the FixInsteadOfThrowAway View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.AdvancedPageItems
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FixInsteadOfThrowAway : ContentPage
    {
        public FixInsteadOfThrowAway()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                AdvancedPointsUpdate helper2 = new AdvancedPointsUpdate();
                helper2.FixPoints();
                GetData data = new GetData();
                data.SetLvl();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.FoodDelivered"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Food Delivered" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.FoodDelivered.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Instead of traveling to the shop have your food delivered 
at once. This is more efficient and will result in less emissions as only a single delivery 
vehicle will have to make a trip to all the houses rather than everyone going there and 
back."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The FoodDelivered View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the FoodDelivered View Class. This class is the eco action that the 
user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FoodDelivered : ContentPage
    {
        public FoodDelivered()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.FoodDelivredPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.FoodInBulk"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Food In Bulk" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Shopping.FoodBulk.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase Food in Bulk to save trips to the store in turn saving energy and reducing emissions."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The FoodInBulk View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the FoodInBulk View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FoodInBulk : ContentPage
    {
        public FoodInBulk()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.FoodInBulkPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.GoCamping"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Go Camping" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.Camping.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Go camping and leave your home for a while. Use as much 
reusable gear as possible and try to use as least energy as possible. This can be quiet 
relaxing. Taking a break from the world."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The GoCamping View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the GoCamping View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class GoCamping : ContentPage
    {
        public GoCamping()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.CampingPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.HangDry"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Hang Dry" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.HangDry.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Instead of using the dryer hang your clothes to dry when 
you can. It may take longer, but it's much better on the environment, in more ways than 
one."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The HangDry View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the HangDry View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class HangDry : ContentPage
    {
        public HangDry()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.HangDryPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.InsulateWater"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Water Tank" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.InsulateWaterTank.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Insulate your water tanks. This will keep the water in them 
warmer for longer periods of time in turn saving you money on electricity."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The InsulateWater View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the InsulateWater View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class InsulateWater : ContentPage
    {
        public InsulateWater()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.InsulateWaterPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.IsolateHome"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Insulate Home" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.InsulateHome.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Insulate your home and keep the heat it. This will not only 
make your home feel nice, hot and cozy but will also in turn save you money on heating."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The IsolateHome View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the IsolateHome View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class IsolateHome : ContentPage
    {
        public IsolateHome()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.IsolateHomePoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.LedLightBulb"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Led Lights" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Energy.LedLight.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Change to LED light bulbs now. They last longer than conventional bulbs and are far more efficient."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The LedLightBulb View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the LedLightBulb View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LedLightBulb : ContentPage
    {
        public LedLightBulb()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.LedLightsPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.LocalProduct"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Buy Local" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.LocalProduct.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Buy things locally if possible and support your community. 
Buying this locally is usually more environmentally friendly as the items you are purchasing 
did not have to be imported. This also will help keep local businesses open."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The LocalProduct View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the LocalProduct View Class. This class is the eco action that the 
user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LocalProduct : ContentPage
    {
        public LocalProduct()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.LocalProductPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.LooseLeafTea"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Loose Leaf" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.LooseTea.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and make loose leaf tea over bagged tea. Bagged 
tea is pretty much pointless but ust more convenient even though it contributes to waste 
with the bags. Loose leaf tea takes the bags out of the equation and even tastes better."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The LooseLeafTea View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the LooseLeafTea View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LooseLeafTea : ContentPage
    {
        public LooseLeafTea()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.TeaPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.MachineFull"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Washing Machine" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.FullWashingMachine.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Only use the washing machine when it is full. This will 
save energy and also a ton of CO2 emission as a Washing Machine emits about 440kg of CO2 a 
year on average."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The MachineFull View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the MachineFull View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MachineFull : ContentPage
    {
        public MachineFull()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.MachineFullPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.MicrowaveNotOven"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Microwave" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.Microwave.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Microwave your food instead of heating it up using the oven 
when you can and save energy. Plus it is faster."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The MicrowaveNotOven View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the MicrowaveNotOven View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MicrowaveNotOven : ContentPage
    {
        public MicrowaveNotOven()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.MicrowavePoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.NoMeat"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="No Meat" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.NoMeat.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Don't eat meat today. This will seriously reduce greenhouse 
gas emissions."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The NoMeat View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the NoMeat View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class NoMeat : ContentPage
    {
        public NoMeat()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.NoMeatPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Home.NonHarmfulProducts"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">
    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Non Harmful" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.NonHarmful.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and use non harmful products. These reduce 
pollution levels and are healthier."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The NonHarmfulProducts View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the NonHarmfulProducts View Class. This class is the eco action that
the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Home
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class NonHarmfulProducts : ContentPage
    {
        public NonHarmfulProducts()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                HomePointsUpdate helper2 = new HomePointsUpdate();
                helper2.NonHarmfulPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Work.OffElectronics"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Electronics Off" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Work.Off.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Turn off electronics that are not in use. There is no need 
to pointlessly waste electricity."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Work
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class OffElectronics : ContentPage
    {
        public OffElectronics()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                WorkPointsUpdate helper2 = new WorkPointsUpdate();
                helper2.ElectonicsOffPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.OffSocketSwitch"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Socket Off" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.OffSocket.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Turn off the wall sockets that are not in use or plug out 
the device if the socket cannot be switched off. This will save energy from the devices 
stand by mode."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The OffSocketSwitch View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the OffSocketSwitch View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class OffSocketSwitch : ContentPage
    {
        public OffSocketSwitch()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.SocketPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.OrganicFood"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Go Organic" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.OrganicFood.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and consume organic food. It contains less 
chemicals and is healthier than non organic food."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The OrganicFood View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the OrganicFood View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class OrganicFood : ContentPage
    {
        public OrganicFood()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.OrganicPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Home.OutsideOnce"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Go Outside" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.Outside.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Go outside once a day and get that Vitamin D!. Going 
outside at least once a day is good for your mental and physical health. In addition no 
electricity is being used by you when you are outside."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="2 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The OutsideOnce View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the OutsideOnce View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Home
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class OutsideOnce : ContentPage
    {
        public OutsideOnce()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTwoPoints();
                HomePointsUpdate helper2 = new HomePointsUpdate();
                helper2.OutsidePoints();
                await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.OwnCoffee"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Brew Coffee" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.OwnCoffee.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchasing coffee and making your own is better for the 
environment that going to the coffee shop."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="2 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The OwnCoffee View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the OwnCoffee View Class. This class is the eco action that the user 
can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class OwnCoffee : ContentPage
    {
        public OwnCoffee()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTwoPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.OwnCoffeePoints();
                await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.Picnic"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Have A Picnic" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.Picnic.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Go out for a picnic and use reusable cutlery. This can be 
very positive for you mood."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The Picnic View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Picnic View Class. This class is the eco action that the user 
can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Picnic : ContentPage
    {
        public Picnic()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.PicnicPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.PlantABush"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Plant A Bush" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.Bush.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Plant a bush where you can. As time goes on the planet is 
loosing more and more of it's greenery. Planting a bush can decrease this loss and it helps 
the environment by producing oxygen."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The PlantABush View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the PlantABush View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PlantABush : ContentPage
    {
        public PlantABush()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.PlantBushPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.PlantAFlower"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Plant A Flower" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.Flower.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Plant flowers where you can. This does not only improve the 
environment but also looks visually appealing and smells nice."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The PlantAFlower View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the PlantAFlower View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PlantAFlower : ContentPage
    {
        public PlantAFlower()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.PlantFlowerPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.PlantATree"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Plant A Tree" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Outdoors.Tree.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Plant a tree where you can. The main environmental effects of deforestation and forest degradation include reduced biodiversity, the release of greenhouse gas emissions, forest fires, disrupted water cycles and increased soil erosion."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The PlantATree View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the PlantATree View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PlantATree : ContentPage
    {
        public PlantATree()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.PlantTreePoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Home.PlantIntoHome"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Plant Inside" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.PlantHome.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Bring a plant into your home. A plant cleans indoor air by 
absorbing toxins, increasing humidity and producing oxygen."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The PlantIntoHome View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the PlantIntoHome View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Home
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PlantIntoHome : ContentPage
    {
        public PlantIntoHome()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                HomePointsUpdate helper2 = new HomePointsUpdate();
                helper2.PlantsInsidePoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Travel.PublicTransport"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Public Transport" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.PublicTransport.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Use public transport when ou cannot cycle or walk as it is 
even more efficient than carpooling or driving."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The PublicTransport View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the PublicTransport View Class. This class is the eco action that 
the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Travel
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PublicTransport : ContentPage
    {
        public PublicTransport()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                TravelPointsUpdate helper2 = new TravelPointsUpdate();
                helper2.TransportPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             
x:Class="Application_Green_Quake.Views.EcoActions.Shopping.PurchaseReusableWater"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Reusable Bottle" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Instead of using and buying plastic bottles use a reusable 
water bottle and refill it each time. This removes your plastic bottle waste. At this rate 
99 million tons of plastic waste will end up in the environment by 2030"
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The PurchaseReusableWater View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the PurchaseReusableWater View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PurchaseReusableWater : ContentPage
    {
        public PurchaseReusableWater()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.ReWaterPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Water.RainBarrel"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Rain Barrel" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Water.RainBarrel.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Set up a rain barrel to collect rain water. This rainwater 
can then bre reused for other thinks such as watering plants."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The RainBarrel View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the RainBarrel View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Water
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RainBarrel : ContentPage
    {
        public RainBarrel()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                WaterPointsUpdate helper2 = new WaterPointsUpdate();
                helper2.BarrelPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.ReBatteries"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Reusable Batteries" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Energy.ReBatteries.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase, use and reuse rechargeable batteries instead of regular batteries. This prevents us from having to make as many as we do and reduces the waste produced by them."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ReBatteries View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ReBatteries View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ReBatteries : ContentPage
    {
        public ReBatteries()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.ReBattereisPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.ReCoffeeMug"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Reusable Cup" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.ReCoffeeMug.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Use a reusable coffee mug or cup for your coffee. This
highly reduces waste produced from throwing away coffee cups. Waste is a large issue as most
of waste does not rot and therefore ends up at a landfill which keep getting bigger and
bigger and this cannot last forever."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ReCoffeeMug View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ReCoffeeMug View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ReCoffeeMug : ContentPage
    {
        public ReCoffeeMug()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.ReCoffeeMugPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.RefrigiratorDown"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Refrigirator" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Energy.TurnDownFridge.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Turn down the refrigerator. Your fridge might be on a higher than needed setting. Turn this down and save energy. The halocarbons in refrigeration appliances contribute to the greenhouse effect which effects the ozone layer and contributes to global warming."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The RefrigiratorDown View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the RefrigiratorDown View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RefrigiratorDown : ContentPage
    {
        public RefrigiratorDown()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.FridgePoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Shopping.ReusableBag"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Reusable Bag" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Shopping.ReBag.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Always use a reusable bag. Using paper or plastic bags for single use add to waste."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```csharp
/*! \class The ReusableBag View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ReusableBag View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Shopping
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ReusableBag : ContentPage
    {
        public ReusableBag()
        {
            InitializeComponent();
            OnAppearing();
        }

        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
                helper2.ReWaterPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Water.ReusableWater"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Reusable Bottle" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Instead of using and buying plastic bottles use a reusable 
water bottle and refill it each time. This removes your plastic bottle waste. At this rate 
99 million tons of plastic waste will end up in the environment by 2030."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```csharp
/*! \class The ReusableWater View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ReusableWater View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Water
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ReusableWater : ContentPage
    {
        public ReusableWater()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                WaterPointsUpdate helper2 = new WaterPointsUpdate();
                helper2.ReWaterPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.SaveLeftOvers"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Save Leftovers" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.SaveLeftovers.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Instead of throwing the leftovers. Save them for later. 
Food waste is a major concern in today world as it is at large. When food rots it emits 
methane, a powerful greenhouse gas which is 25 times worse for the ozone layer than CO2. 60% 
of methane is produced by humans."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SaveLeftOvers View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SaveLeftOvers View Class. This class is the eco action that the 
user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SaveLeftOvers : ContentPage
    {
        public SaveLeftOvers()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.SaveLeftOversPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.Scoop"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Scoop da Poop" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Outdoors.Scoop.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="When you dog makes a mess clean it up. It is your duty as a dog owner"
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The Scoop View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Scoop View Class. This class is the eco action that the user can
log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Scoop : ContentPage
    {
        public Scoop()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.ScoopPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.SealDrafts"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Seal Draft" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.SealDraft.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Seal the drafts in your home and keep it warm. This will 
save you a ton of energy as you will not have to use the heating as frequently."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SealDrafts View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SealDrafts View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SealDrafts : ContentPage
    {
        public SealDrafts()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.SealDraftsPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.SealDucts"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Seal Duct" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.SealDuct.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Seal the ducts in your home and keep it warm. This will 
save you a ton of energy as you will not have to use the heating as frequently."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SealDucts View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SealDucts View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SealDucts : ContentPage
    {
        public SealDucts()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.SealDuctsPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.SetUpFruitGarden"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Fruit Garden" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.FruitGarden.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Set up a fruit garden. This is great for the environment 
and makes you self sustainable and you will no longer need to purchase some fruits."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SetUpFruitGarden View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SetUpFruitGarden View Class. This class is the eco action that 
the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SetUpFruitGarden : ContentPage
    {
        public SetUpFruitGarden()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.FruitGardenPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.SetUpHerbGarden"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Herb Garden" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.HerbGarden.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Set up a Herb garden. This is great for the environment and 
makes you self sustainable and you will no longer need to purchase some herbs."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SetUpHerbGarden View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SetUpHerbGarden View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SetUpHerbGarden : ContentPage
    {
        public SetUpHerbGarden()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.HerbGardenPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Waste.SetUpRecyclingBin"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Recycling Bins" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.SetUpBins.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Set up recycling bins in you home. This reduces the amount 
of waste sent to landfills and incinerators. Conserves natural resources such as timber, 
water and minerals and saves energy."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SetUpRecyclingBin View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SetUpRecyclingBin View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Waste
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SetUpRecyclingBin : ContentPage
    {
        public SetUpRecyclingBin()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                WastePointsUpdate helper2 = new WastePointsUpdate();
                helper2.SetUpRecyclingBinPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.SetUpVegetableGarden"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Vegetable Garden" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.VegGarden.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Set up a Vegetable garden. This is great for the
environment and makes you self sustainable and you will no longer need to purchase some
vegetables."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SetUpVegetableGarden View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SetUpVegetableGarden View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SetUpVegetableGarden : ContentPage
    {
        public SetUpVegetableGarden()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.VegetableGardenPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Community.ShareThisApp"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Share App" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Community.Share.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="This one is simple! Share this app and let's grow 
together!"
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ShareThisApp View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ShareThisApp View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Community
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ShareThisApp : ContentPage
    {
        public ShareThisApp()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
                helper2.SharePoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Water.ShowerBucket"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Shower Bucket" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Water.ShowerBucket.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Place a shower bucket in your shower to collect the water 
you use. This water can be used for other things such as watering plants"
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ShowerBucket View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ShowerBucket View Class. This class is the eco action that the
user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Water
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ShowerBucket : ContentPage
    {
        public ShowerBucket()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                WaterPointsUpdate helper2 = new WaterPointsUpdate();
                helper2.ShowerBucketPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Habits.ShowerInstead"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Shower No Bath" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.ShowerNoBath.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Shower instead of taking a bath. This is more efficient. If 
you don't take too long of course. The average bath uses 36 gallons to fill a tub, while the 
average shower (without the water-saving device) uses five gallons of water per minute."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="6 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ShowerInstead View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ShowerInstead View Class. This class is the eco action that the 
user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Habits
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ShowerInstead : ContentPage
    {
        public ShowerInstead()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateBySixPoints();
                HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
                helper2.ShowerInsteadPoints();
                await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Energy.SolarPanel"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Solar Panel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Energy.SolarPanel.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Install a Solar Panel. Using solar energy can have a positive, indirect effect on the environment when solar energy replaces or reduces the use of other energy sources that have larger effects on the environment."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SolarPanel View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SolarPanel View Class. This class is the eco action that the 
user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Energy
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SolarPanel : ContentPage
    {
        public SolarPanel()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security 
methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per 
day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the 
next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
                helper2.SolarPanelPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Community.SpreadAwareness"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Spread Awereness" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Community.Spread.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Spread Awareness about the environment by posting online 
and the alike. 77% of people would like to live more sustainably and you could be helping 
them by showing how."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SpreadAwareness View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SpreadAwareness View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Community
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SpreadAwareness : ContentPage
    {
        public SpreadAwareness()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
                helper2.awarenessPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.SteelStraw"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Steel Straws" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.SteeStraw.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Purchase and use steel straws over regular straws. This
means you can reuse them and reduces waste. In just the U.S. alone, one estimate suggests
500 million straws are used every single day."
                       FontSize="20"
                       TextColor="Black"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The SteelStraw View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the SteelStraw View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SteelStraw : ContentPage
    {
        public SteelStraw()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.SteelStrawPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Habits.TimedShower"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Timed Shower" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.TimedShower.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Time your showers. A shower uses five gallons of water per
minute when it does not have a water saving shower head. Therefore try spending a max of 3
minutes in there."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The TimedShower View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the TimedShower View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Habits
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TimedShower : ContentPage
    {
        public TimedShower()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
                helper2.TimedShowerInsteadPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Home.ToiletFlushes"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Save A Flush" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.Flush.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Instead of flushing the toilet every time try and flush 
only when it is necessary to save water."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ToiletFlushes View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ToiletFlushes View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Home
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ToiletFlushes : ContentPage
    {
        public ToiletFlushes()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                HomePointsUpdate helper2 = new HomePointsUpdate();
                helper2.ToiletPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Habits.TurnOffLights"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Turn Off Lights" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.OffLights.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Turn off the lights when leaving the room. If you turn off 
the lights whenever you leave a room, you can reduce greenhouse gas emissions by 0.15 pounds 
per hour."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="2 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The TurnOffLights View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the TurnOffLights View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Habits
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TurnOffLights : ContentPage
    {
        public TurnOffLights()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTwoPoints();
                HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
                helper2.OffLightsPoints();
                await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.UpBirdfeeder"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Bird Feeder" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.BirdFeeder.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Set up a bird feeder and help the birds. Support 
biodiversity by supporting other animals"
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The UpBirdfeeder View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the UpBirdfeeder View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Outdoors
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class UpBirdfeeder : ContentPage
    {
        public UpBirdfeeder()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
                helper2.BirdFeederPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Waste.UseBiogradableBinBags"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Bio Bin Bags" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" 
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.BioBags.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Use biodegradable bin bags as these can rot and don't 
pollute the environment."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="4 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The UseBiogradableBinBags View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the UseBiogradableBinBags View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Waste
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class UseBiogradableBinBags : ContentPage
    {
        public UseBiogradableBinBags()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByFourPoints();
                WastePointsUpdate helper2 = new WastePointsUpdate();
                helper2.BioBinBagPoints();
                await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Habits.UseMatches"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Matches" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.MatchOverLighter.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Use mathces over lighters. Matches are biodegradable , 
whereas lighters are plastic and aren't recyclable in most places. The creation of lighters 
also involves more processing and more transportation of the various components, which come 
with their own waste and pollution."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="2 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The UseMatches View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the UseMatches View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Habits
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class UseMatches : ContentPage
    {
        public UseMatches()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTwoPoints();
                HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
                helper2.MatchesPoints();
                await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Waste.UseRecyclingBin"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Recycle" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Waste.Recycled.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Recycle when you can! This reduces the amount of waste sent
to landfills and incinerators. Conserves natural resources such as timber, water and
minerals and saves energy."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The UseRecyclingBin View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the UseRecyclingBin View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Waste
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class UseRecyclingBin : ContentPage
    {
        public UseRecyclingBin()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                WastePointsUpdate helper2 = new WastePointsUpdate();
                helper2.RecyclingBinPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Travel.Walk"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Walk" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.Walk.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Walk there if you can as this is the most environmentally 
friendly from of transportation."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The Walk View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Walk View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Travel
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Walk : ContentPage
    {
        public Walk()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                TravelPointsUpdate helper2 = new TravelPointsUpdate();
                helper2.WalkPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.WaterOverFizzy"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="H2O" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.WaterOverFiz.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Drink water instead of other drinks. This is much healthier 
and better for the environment. Water is the healthiest drink for the human body and is 
better for the environment as it takes a lot more to produce fizzy drinks which emit CO2."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="8 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The WaterOverFizzy View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WaterOverFizzy View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WaterOverFizzy : ContentPage
    {
        public WaterOverFizzy()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByEightPoints();
                FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
                helper2.WaterOverFizzyPoints();
                await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Work.WorkingRemotely"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Remote Work" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Work.Remote.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="If possible work remotely. Not only is this more relaxing and less time consuming but it also saves travel costs and reduces emissions."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Work
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WorkingRemotely : ContentPage
    {
        public WorkingRemotely()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                WorkPointsUpdate helper2 = new WorkPointsUpdate();
                helper2.RemoteWorkPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.Water.WSShowerHead"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="WSS Head" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="2*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <Image Aspect="AspectFill"
                   Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Water.ShowerHead.jpg}"/>
                <StackLayout Grid.Row="1"
                         VerticalOptions="Center"
                         HorizontalOptions="Center"
                         Margin="15,0,15,0"
                         Padding="0,5,0,5"
                         BackgroundColor="#D3D3D3">
                    <Label Text="Install a Water saving Shower Head to use less water when 
you shower."
                       TextColor="Black"
                       HorizontalTextAlignment="Center"
                       FontSize="20"/>
                </StackLayout>
                <StackLayout Grid.Row="2"
                         Spacing="10">
                    <Label Text="10 POINTS!"
                       TextColor="Black"
                       FontAttributes="Bold"
                       FontSize="20"
                       HorizontalOptions="Center"/>
                    <Button Text="Completed"
                        BackgroundColor="#50C878"
                        TextColor="White"
                        Margin="60,0,60,0"
                        Clicked="AddPointsClicked"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The WSShowerHead View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WSShowerHead View Class. This class is the eco action that the user can log.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.Water
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WSShowerHead : ContentPage
    {
        public WSShowerHead()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
          * methods
        */
        private async void AddPointsClicked(object sender, EventArgs e)
        {
            SecurityMethods checks = new SecurityMethods();
            Task<bool> myTask = checks.DayLimitLock();
            await myTask;

            Task<bool> myTaskTwo = checks.TimeLimitLock();
            await myTaskTwo;

            if (myTask.Result)
            {
                await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per day.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else if (myTaskTwo.Result)
            {
                await DisplayAlert("Too soon", "You must wait 1 minute before logging the next Action.", "OK");
                await Navigation.PushAsync(new MainMenu());
            }
            else
            {
                PointsUpdate helper = new PointsUpdate();
                helper.UpdateByTenPoints();
                WaterPointsUpdate helper2 = new WaterPointsUpdate();
                helper2.WSSowerHeadPoints();
                await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
                await Navigation.PushAsync(new MainMenu());
            }
        }
        /** This function creates objects and calls their methods. First the security methods are called and if they return false call the points updating
         * methods
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<pages:PopupPage x:Class="Application_Green_Quake.Views.LeaderboardPage.LeaderBoardPopUp"
                 xmlns="http://xamarin.com/schemas/2014/forms"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 xmlns:animations="clr-
namespace:Rg.Plugins.Popup.Animations;assembly=Rg.Plugins.Popup"
                 xmlns:pages="clr-
namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
                 BackgroundColor="#002a1e">

    <StackLayout HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" >
        <Label x:Name="theName" TextColor="White" FontSize="20"
HorizontalTextAlignment="Center"/>
        <Image x:Name="profileImage" HeightRequest="300"/>
        <Label x:Name="theRank" TextColor="White" FontSize="20"
HorizontalTextAlignment="Center"/>
        <Label x:Name="theBio" TextColor="White" FontSize="20"
HorizontalTextAlignment="Center"/>
        <Label x:Name="thePoints" TextColor="White" FontSize="20"
HorizontalTextAlignment="Center"/>
    </StackLayout>

</pages:PopupPage>
```

```
/*! \class The LeaderBoardPopUp View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the LeaderBoardPopUp View Class. This class is the popup that appears when a entry in the LeaderBoard is tapped.
 *
 */
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.LeaderboardPage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LeaderBoardPopUp
    {
        /** The LeaderBoardPopUp Constructor
        @param username is the username to display
        @param points are the points to display
        @param rank is the rank to display
        @param image is the image to display
        @param bio is the bio to display
        */
        public LeaderBoardPopUp(string username, int points, string rank, ImageSource image, string bio)
        {
            InitializeComponent();
            profileImage.Source = image;
            theName.Text = "Username: " + username;
            thePoints.Text = "Points: " + points.ToString();
            theBio.Text = "Bio: " + bio;
            theRank.Text = rank;
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:models="clr-namespace:Application_Green_Quake.Models;assembly=Application Green Quake"
             xmlns:xForms="clr-namespace:Syncfusion.SfPicker.XForms;assembly=Syncfusion.SfPicker.XForms"
             x:Class="Application_Green_Quake.Views.LeaderboardPage.TopTabLeaderBoard">
    <ContentPage.Content>
        <StackLayout BackgroundColor="#002a1e">
            <xForms:SfPicker x:Name="picker"
                             ItemHeight="45"
                             HeaderText="Filter by Country"
                             PickerMode="Dialog"
                             PickerHeight="350"
                             PickerWidth="350"
                             ShowFooter="True"
                             OkButtonClicked="PickerOnOkButtonClicked"
                             HeaderBackgroundColor="#50C878"
                             SelectedItemTextColor="#50C878"
                             OKButtonTextColor="#50C878"
                             CancelButtonTextColor="#50C878"/>
            <StackLayout Orientation="Horizontal" Margin="15,5,15,5">
                <Label Text="Leader Board" FontSize="22" HorizontalTextAlignment="Center" FontFamily="Proxima Nova" TextColor="White"/>
                <Image Source="{models:ImageResource Application_Green_Quake.Images.filter.png}" HorizontalOptions="EndAndExpand">
                    <Image.GestureRecognizers>
                        <TapGestureRecognizer Tapped="ImageClicked"/>
                    </Image.GestureRecognizers>
                </Image>
            </StackLayout>
            <ListView x:Name="LeaderBoard"
                      ItemTapped="OnItemTapped"
                      RowHeight="70">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout BackgroundColor="White">
                                <Grid Margin="20,0,20,0">
                                    <Grid.ColumnDefinitions>
                                        <ColumnDefinition Width="*"/>
                                        <ColumnDefinition Width="2*"/>
                                        <ColumnDefinition Width="2*"/>
                                    </Grid.ColumnDefinitions>
                                    <Grid.RowDefinitions>
                                        <RowDefinition Height="*"/>
                                    </Grid.RowDefinitions>

                                    <Frame Padding="0" Margin="0,5,0,5" CornerRadius="80">
                                        <Image Source="{Binding image}" Aspect="AspectFill"/>
                                    </Frame>

                                    <StackLayout Grid.Column="1" Margin="10,5,0,0" Spacing="0">
                                        <Label Text="{Binding username}"  TextColor="Black" FontAttributes="Bold" FontSize="15" FontFamily="Roboto"/>
                                        <Label Text="{Binding points, StringFormat='Score: {0}'}" FontSize="12" TextColor="Silver" FontFamily="Roboto"/>
                                    </StackLayout>
                                    <StackLayout Grid.Column="2">
```

```
                                        <Label Text="{Binding rank}" Margin="0,5,0,0" HorizontalOptions="End" FontFamily="Roboto" TextColor="Silver"/>
                                    </StackLayout>
                                </Grid>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
            <StackLayout Orientation="Horizontal" Margin="15,5,15,5">
                <Label Text="Back To Page 1" FontSize="22" FontFamily="Proxima Nova" TextColor="White" HorizontalOptions="Start" HeightRequest="30">
                    <Label.GestureRecognizers>
                        <TapGestureRecognizer Tapped="FirstPageClicked"/>
                    </Label.GestureRecognizers>
                </Label>
                <Image Source="{models:ImageResource Application_Green_Quake.Images.right.png}" HorizontalOptions="EndAndExpand" HeightRequest="30">
                    <Image.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NextPageClicked"/>
                    </Image.GestureRecognizers>
                </Image>
            </StackLayout>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The TopTabLeaderBoard View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the TopTabLeaderBoard View Class. This is the class for the leaderboard screen.
 *
 */
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using Firebase.Storage;
using Rg.Plugins.Popup.Services;
using System;
using System.Collections.ObjectModel;
using System.Linq;
using Acr.UserDialogs;
using Application_Green_Quake.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using SelectionChangedEventArgs = Syncfusion.SfPicker.XForms.SelectionChangedEventArgs;

namespace Application_Green_Quake.Views.LeaderboardPage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TopTabLeaderBoard : ContentPage
    {
        private string selectedNation = "All";
        private int min = 0;
        private int count = 10;
        IAuth auth;
        public TopTabLeaderBoard()
        {
            InitializeComponent();
            ObservableCollection<object> countryNameCollection = new ObservableCollection<object>();

            // Add al the options into the filter
            countryNameCollection.Add("All");
            countryNameCollection.Add("Me");
            countryNameCollection.Add("Albania");
            countryNameCollection.Add("Andorra");
            countryNameCollection.Add("Armenia");
            countryNameCollection.Add("Austria");
            countryNameCollection.Add("Azerbaijan");
            countryNameCollection.Add("Belarus");
            countryNameCollection.Add("Belgium");
            countryNameCollection.Add("Bosnia and Herzegovina");
            countryNameCollection.Add("Bulgaria");
            countryNameCollection.Add("Croatia");
            countryNameCollection.Add("Cyprus");
            countryNameCollection.Add("Czech Republic");
            countryNameCollection.Add("Denmark");
            countryNameCollection.Add("Estonia");
            countryNameCollection.Add("Finland");
            countryNameCollection.Add("France");
            countryNameCollection.Add("Georgia");
            countryNameCollection.Add("Germany");
            countryNameCollection.Add("Greece");
            countryNameCollection.Add("Hungary");
            countryNameCollection.Add("Iceland");
            countryNameCollection.Add("Ireland");
            countryNameCollection.Add("Italy");
            countryNameCollection.Add("Kazakhstan");
            countryNameCollection.Add("Kosovo");
            countryNameCollection.Add("Latvia");
            countryNameCollection.Add("Liechtenstein");
            countryNameCollection.Add("Lithuania");
            countryNameCollection.Add("Luxembourg");
            countryNameCollection.Add("Malta");
            countryNameCollection.Add("Moldova");
            countryNameCollection.Add("Monaco");
            countryNameCollection.Add("Montenegro");
            countryNameCollection.Add("Netherlands");
            countryNameCollection.Add("North Macedonia");
            countryNameCollection.Add("Norway");
            countryNameCollection.Add("Poland");
            countryNameCollection.Add("Portugal");
            countryNameCollection.Add("Romania");
            countryNameCollection.Add("Russia");
            countryNameCollection.Add("San Marino");
            countryNameCollection.Add("Serbia");
            countryNameCollection.Add("Slovakia");
            countryNameCollection.Add("Slovenia");
            countryNameCollection.Add("Spain");
            countryNameCollection.Add("Sweden");
            countryNameCollection.Add("Switzerland");
            countryNameCollection.Add("Turkey");
            countryNameCollection.Add("Ukraine");
            countryNameCollection.Add("United Kingdom");
            countryNameCollection.Add("Vatican City");

            picker.ItemsSource = countryNameCollection;
            auth = DependencyService.Get<IAuth>();
            OnAppearing();
        }
        /** This function is called before the page is displayed.
        */
        protected override async void OnAppearing()
        {
            //If All gets selected from the filter then display all the profiles
            if (selectedNation == "All")
            {
                UserDialogs.Instance.ShowLoading();
                FirebaseClient firebaseClient = new FirebaseClient("https://application-
green-quake-default-rtdb.firebaseio.com/");

                //Save the data into a list and order it by points
                var list = (await firebaseClient
                      .Child("Points")
                      .OnceAsync<Points>()).Select(item => new Points
                      {
                          username = item.Object.username,
                          points = item.Object.points,
                      }).ToList().OrderByDescending(s => s.points);

                //Save that data to a second list
                var list2 = list.Select(item => new LeaderBoard
                {
                    username = item.username,
                    points = item.points,
                }).ToList();

                int index = 0;
                string rankIndex = "";
                // Create a new list and assign an image and the rank based on the index
                foreach (var i in list2)
                {
                    index++;
                    rankIndex = "Rank: " + index.ToString();
                    i.rank = rankIndex;
                    try
                    {
                        var uid = (await firebaseClient
                        .Child("usernames")
                        .Child(i.username)
                        .OnceSingleAsync<Usernames>()).Uid;

                        i.image = await new FirebaseStorage("application-green-
quake.appspot.com")
                        .Child(uid)
                        .Child("Profile.jpg")
                        .GetDownloadUrlAsync();
                    }
                    catch (Exception e)
                    {
                        i.image =
ImageSource.FromResource("Application_Green_Quake.Images.user.png");
                        Console.Write(e);
                    }
                    try
                    {
                        var uid = (await firebaseClient
                        .Child("usernames")
                        .Child(i.username)
                        .OnceSingleAsync<Usernames>()).Uid;

                        i.bio = (await firebaseClient
                        .Child("users")
                        .Child(uid)
                        .OnceSingleAsync<Users>()).bio;

                        i.nation = (await firebaseClient
                            .Child("users")
                            .Child(uid)
                            .OnceSingleAsync<Users>()).nation;
                    }
                    catch (Exception e)
                    {
                        Console.Write(e);
                    }
                }

                try
                {   //Get entries between specified indexes and dsave them into a list and
then display them in the leaderboard.
                    var list3 = list2.Select(item => new LeaderBoard
                    {
                        username = item.username,
                        points = item.points,
                        nation = item.nation,
                        bio = item.nation,
                        rank = item.rank,
                        image = item.image
                    }).ToList().GetRange(min, count);
                    LeaderBoard.ItemsSource = list3;

                }
                catch (Exception e)
                {
                    try
                    {   //Get entries between specified indexes and dsave them into a list
and then display them in the leaderboard.
                        var list3 = list2.Select(item => new LeaderBoard
                        {
                            username = item.username,
                            points = item.points,
                            nation = item.nation,
                            bio = item.bio,
                            rank = item.rank,
                            image = item.image
                        }).ToList().GetRange(min, list2.Count - min);
                        LeaderBoard.ItemsSource = list3;
                    }
                    catch (Exception exception)
                    {
                        min = 0;
                        await DisplayAlert("Last Page", "You have reached the last leader
board page.", "Ok");
                    }
                }
                UserDialogs.Instance.HideLoading();
            }
            //If Me gets selected from the filter then only display the users profile
            else if (selectedNation == "Me")
            {
                UserDialogs.Instance.ShowLoading();
                FirebaseClient firebaseClient = new FirebaseClient("https://application-
green-quake-default-rtdb.firebaseio.com/");

                var list = (await firebaseClient
                      .Child("Points")
                      .OnceAsync<Points>()).Select(item => new Points
                      {
                          username = item.Object.username,
                          points = item.Object.points,
                      }).ToList().OrderByDescending(s => s.points);

                var list2 = list.Select(item => new LeaderBoard
                {
                    username = item.username,
                    points = item.points,
                }).ToList();

                int index = 0;
                string rankIndex = "";
                foreach (var i in list2)
                {
                    index++;
                    rankIndex = "Rank: " + index.ToString();
                    i.rank = rankIndex;
                    try
                    {
                        var uid = (await firebaseClient
                        .Child("usernames")
                        .Child(i.username)
                        .OnceSingleAsync<Usernames>()).Uid;

                        i.uid = uid;

                        i.image = await new FirebaseStorage("application-green-
quake.appspot.com")
                        .Child(uid)
                        .Child("Profile.jpg")
                        .GetDownloadUrlAsync();
                    }
                    catch (Exception e)
                    {
                        i.image =
ImageSource.FromResource("Application_Green_Quake.Images.user.png");
                        Console.Write(e);
                    }
                    try
                    {
                        var uid = (await firebaseClient
                        .Child("usernames")
                        .Child(i.username)
                        .OnceSingleAsync<Usernames>()).Uid;

                        i.uid = uid;

                        i.bio = (await firebaseClient
                        .Child("users")
                        .Child(uid)
                        .OnceSingleAsync<Users>()).bio;

                        i.nation = (await firebaseClient
                            .Child("users")
                            .Child(uid)
                            .OnceSingleAsync<Users>()).nation;
                    }
                    catch (Exception e)
                    {
                        Console.Write(e);
                    }
                }

                var list3 = list2.Select(item => new LeaderBoard
                {
                    username = item.username,
                    points = item.points,
                    nation = item.nation,
                    bio = item.nation,
                    rank = item.rank,
                    image = item.image,
                    uid = item.uid
                }).Where(x => x.uid == auth.GetUid()).ToList();

                LeaderBoard.ItemsSource = list3;
                UserDialogs.Instance.ShowLoading();
            }
            else
            {
                UserDialogs.Instance.ShowLoading();
                FirebaseClient firebaseClient = new FirebaseClient("https://application-
green-quake-default-rtdb.firebaseio.com/");
```

```

                var list = (await firebaseClient
                      .Child("Points")
                      .OnceAsync<Points>()).Select(item => new Points
                      {
                          username = item.Object.username,
                          points = item.Object.points,
                      }).ToList().OrderByDescending(s => s.points);

                var list2 = list.Select(item => new LeaderBoard
                {
                    username = item.username,
                    points = item.points,
                }).ToList();

                int index = 0;
                string rankIndex = "";
                foreach (var i in list2)
                {
                    index++;
                    rankIndex = "Rank: " + index.ToString();
                    i.rank = rankIndex;
                    try
                    {
                        var uid = (await firebaseClient
                        .Child("usernames")
                        .Child(i.username)
                        .OnceSingleAsync<Usernames>()).Uid;

                        i.image = await new FirebaseStorage("application-green-
quake.appspot.com")
                        .Child(uid)
                        .Child("Profile.jpg")
                        .GetDownloadUrlAsync();
                    }
                    catch (Exception e)
                    {
                        i.image =
ImageSource.FromResource("Application_Green_Quake.Images.user.png");
                        Console.Write(e);
                    }
                    try
                    {
                        var uid = (await firebaseClient
                        .Child("usernames")
                        .Child(i.username)
                        .OnceSingleAsync<Usernames>()).Uid;

                        i.bio = (await firebaseClient
                        .Child("users")
                        .Child(uid)
                        .OnceSingleAsync<Users>()).bio;

                        i.nation = (await firebaseClient
                            .Child("users")
                            .Child(uid)
                            .OnceSingleAsync<Users>()).nation;
                    }
                    catch (Exception e)
                    {
                        Console.Write(e);
                    }
                }

                var list3 = list2.Select(item => new LeaderBoard
                {
                    username = item.username,
                    points = item.points,
                    nation = item.nation,
                    bio = item.bio,
                    rank = item.rank,
                    image = item.image
                }).Where(n => n.nation == selectedNation).ToList();

                index = 0;
                rankIndex = "";
                foreach (var p in list3)
                {
                    index++;
                    rankIndex = "Rank: " + index.ToString();
                    p.rank = rankIndex;
                }

                LeaderBoard.ItemsSource = list3;
            }
            //Stop the loading spinner and set the data.
            UserDialogs.Instance.HideLoading();
            GetData data = new GetData();
            data.SetLvl();
            min = 0;
            count = 10;
        }

        /** This function displays the correct popup when a profile on the leader board is tapped.
        */
        private void OnItemTapped (Object sender, ItemTappedEventArgs e)
        {

            var dataItem = e.Item as LeaderBoard;
            PopupNavigation.Instance.PushAsync(new LeaderBoardPopUp(dataItem.username, dataItem.points, dataItem.rank, dataItem.image, dataItem.bio));
        }
        /** This function displays the picker when the icon is pressed.
        */
        private async void ImageClicked(object sender, EventArgs e)
        {
            picker.IsOpen = true;
        }

        /** This function sets the selectedNation variable when a value is selected on the picker and ok is pressed..
        */
        private async void PickerOnOkButtonClicked(object sender, SelectionChangedEventArgs e)
        {
            if (picker.SelectedItem.ToString() == "All")
            {
                selectedNation = "All";
            }
            else if (picker.SelectedItem.ToString() == "Me")
            {
                selectedNation = "Me";
            }
            else if (picker.SelectedItem.ToString() == "Albania")
            {
                selectedNation = "Albania";
            }
            else if (picker.SelectedItem.ToString() == "Andorra")
            {
                selectedNation = "Andorra";
            }
            else if (picker.SelectedItem.ToString() == "Armenia")
            {
                selectedNation = "Armenia";
            }
            else if (picker.SelectedItem.ToString() == "Austria")
            {
                selectedNation = "Austria";
            }
            else if (picker.SelectedItem.ToString() == "Azerbaijan")
            {
                selectedNation = "Azerbaijan";
            }
            else if (picker.SelectedItem.ToString() == "Belarus")
            {
                selectedNation = "Belarus";
            }
            else if (picker.SelectedItem.ToString() == "Belgium")
            {
                selectedNation = "Belgium";
            }
            else if (picker.SelectedItem.ToString() == "Bosnia and Herzegovina")
            {
                selectedNation = "Bosnia and Herzegovina";
            }
            else if (picker.SelectedItem.ToString() == "Bulgaria")
            {
                selectedNation = "Bulgaria";
            }
            else if (picker.SelectedItem.ToString() == "Cyprus")
            {
                selectedNation = "Cyprus";
            }
            else if (picker.SelectedItem.ToString() == "Czech Republic")
            {
                selectedNation = "Czech Republic";
            }
            else if (picker.SelectedItem.ToString() == "Denmark")
            {
                selectedNation = "Denmark";
            }
            else if (picker.SelectedItem.ToString() == "Estonia")
            {
                selectedNation = "Estonia";
            }
            else if (picker.SelectedItem.ToString() == "Finland")
            {
                selectedNation = "Finland";
            }
            else if (picker.SelectedItem.ToString() == "France")
            {
                selectedNation = "France";
            }
            else if (picker.SelectedItem.ToString() == "Georgia")
            {
                selectedNation = "Georgia";
            }
            else if (picker.SelectedItem.ToString() == "Germany")
            {
                selectedNation = "Germany";
            }
            else if (picker.SelectedItem.ToString() == "Greece")
            {
                selectedNation = "Greece";
            }
            else if (picker.SelectedItem.ToString() == "Hungary")
            {
                selectedNation = "Hungary";
            }
            else if (picker.SelectedItem.ToString() == "Iceland")
            {
                selectedNation = "Iceland";
            }
            else if (picker.SelectedItem.ToString() == "Ireland")
            {
                selectedNation = "Ireland";
            }
            else if (picker.SelectedItem.ToString() == "Italy")
            {
                selectedNation = "Italy";
            }
            else if (picker.SelectedItem.ToString() == "Kazakhstan")
            {
                selectedNation = "Kazakhstan";
            }
            else if (picker.SelectedItem.ToString() == "Kosovo")
            {
                selectedNation = "Kosovo";
            }
            else if (picker.SelectedItem.ToString() == "Latvia")
            {
                selectedNation = "Latvia";
            }
            else if (picker.SelectedItem.ToString() == "Liechtenstein")
            {
                selectedNation = "Liechtenstein";
            }
            else if (picker.SelectedItem.ToString() == "Lithuania")
            {
                selectedNation = "Lithuania";
            }
            else if (picker.SelectedItem.ToString() == "Luxembourg")
            {
                selectedNation = "Luxembourg";
            }
            else if (picker.SelectedItem.ToString() == "Malta")
            {
                selectedNation = "Malta";
            }
            else if (picker.SelectedItem.ToString() == "Moldova")
            {
                selectedNation = "Moldova";
            }
            else if (picker.SelectedItem.ToString() == "Monaco")
            {
                selectedNation = "Monaco";
            }
            else if (picker.SelectedItem.ToString() == "Montenegro")
            {
                selectedNation = "Montenegro";
            }
            else if (picker.SelectedItem.ToString() == "Netherlands")
            {
                selectedNation = "Netherlands";
            }
            else if (picker.SelectedItem.ToString() == "North Macedonia")
            {
                selectedNation = "North Macedonia";
            }
            else if (picker.SelectedItem.ToString() == "Norway")
            {
                selectedNation = "Norway";
            }
            else if (picker.SelectedItem.ToString() == "Poland")
            {
                selectedNation = "Poland";
            }
            else if (picker.SelectedItem.ToString() == "Portugal")
            {
                selectedNation = "Portugal";
            }
            else if (picker.SelectedItem.ToString() == "Romania")
            {
                selectedNation = "Romania";
            }
            else if (picker.SelectedItem.ToString() == "Russia")
            {
                selectedNation = "Russia";
            }
            else if (picker.SelectedItem.ToString() == "San Marino")
            {
                selectedNation = "San Marino";
            }
            else if (picker.SelectedItem.ToString() == "Serbia")
            {
                selectedNation = "Serbia";
            }
            else if (picker.SelectedItem.ToString() == "Slovakia")
            {
                selectedNation = "Slovakia";
            }
            else if (picker.SelectedItem.ToString() == "Slovenia")
            {
                selectedNation = "Albania";
            }
            else if (picker.SelectedItem.ToString() == "Spain")
            {
                selectedNation = "Spain";
            }
            else if (picker.SelectedItem.ToString() == "Sweden")
            {
                selectedNation = "Sweden";
            }
            else if (picker.SelectedItem.ToString() == "Switzerland")
            {
                selectedNation = "Switzerland";
            }
            else if (picker.SelectedItem.ToString() == "Turkey")
            {
                selectedNation = "Turkey";
            }
            else if (picker.SelectedItem.ToString() == "Ukraine")
            {
                selectedNation = "Ukraine";
            }
            else if (picker.SelectedItem.ToString() == "United Kingdom")
            {
                selectedNation = "United Kingdom";
            }
            else if (picker.SelectedItem.ToString() == "Vatican City")
            {
                selectedNation = "Vatican City";
            }

            OnAppearing();
        }

        /** This function displays the next ten items on the leader board.
        */
        private void NextPageClicked(object sender, EventArgs e)
        {
            min = min + 10;
            OnAppearing();
        }

        /** This function displays the first page of the leader board again.
        */
        private void FirstPageClicked(object sender, EventArgs e)
        {
            min = 0;
            OnAppearing();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.AdvancedPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Advanced" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Advanced.Fix.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToFix"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToFix"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Fix It"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Instead of throwing it away try fix it."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The AdvancedPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the AdvancedPage View Class. This page displays and allows the navigation to each of the actions in the Advanced category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.AdvancedPageItems;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AdvancedPage : ContentPage
    {
        public AdvancedPage()
        {
            InitializeComponent();
            OnAppearing();
        }

        /** This function navigates to FixInsteadOfThrowAway.
        */
        private async void NavigateToFix(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new FixInsteadOfThrowAway());
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
          * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.CommunityPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Community Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Community" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.JoinE.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEnvironmentalGroups"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEnvironmentalGroups"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Join An Environmental Group"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"
                           HorizontalTextAlignment="Center"/>
                    <Label Text="Join an environmental group and do your part."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.CreateE.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCreateEnvironmentalGroup"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCreateEnvironmentalGroup"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Create An Environmental Group"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"
                           HorizontalTextAlignment="Center"/>
                    <Label Text="No environmental group to join? No problem!"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.Spread.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSpreadAwareness"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSpreadAwareness"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Spread Awereness"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Spread awareness about environmental issues."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.DoCommunity.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDoCommunity"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDoCommunity"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Do Something For The Community"
                           TextColor="White"
                           HorizontalOptions="Center"
                           HorizontalTextAlignment="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Do something for the community and improve peoples lives."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.Donate.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDonateItems"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDonateItems"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Donate Something"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Donate an item instead of throwing it away or storing it 
forever."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="5"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Community.Share.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToShareThisaApp"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToShareThisaApp"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Share This App"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Share this app and lets grow together."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
```

```
                                 HorizontalTextAlignment="Center"/>
                    </StackLayout>
                </Grid>
            </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The CommunityPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the CommunityPage View Class. This page displays and allows the 
navigation to each of the actions in the Community category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Community;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CommunityPage : ContentPage
    {
        public CommunityPage()
        {
            InitializeComponent();
            OnAppearing();
        }

        /** This function navigates to EnvironmentalGroups.
        */
        private async void NavigateToEnvironmentalGroups(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EnvironmentalGroups());
        }
        /** This function navigates to CreateEnvironmentalGroup.
        */
        private async void NavigateToCreateEnvironmentalGroup(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new CreateEnvironmentalGroup());
        }
        /** This function navigates to SpreadAwareness.
        */
        private async void NavigateToSpreadAwareness(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SpreadAwareness());
        }
        /** This function navigates to DoCommunity.
        */
        private async void NavigateToDoCommunity(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new DoCommunity());
        }
        /** This function navigates to DonateItems.
        */
        private async void NavigateToDonateItems(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new DonateItems());
        }
        /** This function navigates to ShareThisApp.
        */
        private async void NavigateToShareThisaApp(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ShareThisApp());
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.EnergyPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Energy Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Energy" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.FullDryer.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDryerFull"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDryerFull"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Full Dryer"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Only use the Dryer when it is full."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.FullWashingMachine.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToMachineFull"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToMachineFull"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Full Washing Machine"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Only use the washing machine when it is full."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.FullDishwasher.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
```

```
                        <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Full Dishwasher"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Only use the dishwasher when it is full."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.HangDry.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToHangDry"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToHangDry"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Hang Dry Clothes"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Instead of using the washing machine. Hang dry your 
clothes instead."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.Thermostat.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEfficientThermostat"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEfficientThermostat"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Efficient Thermostat"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Efficiently program your thermostat."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="5"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.SolarPanel.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSolarPanel"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSolarPanel"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Solar Energy"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Install a solar panel on your home or garden."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="6"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.LedLight.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToLedLightbulb"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToLedLightbulb"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Led Lights"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Install Led Light Bulbs instead of normal ones."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="7"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.OffSocket.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOffSocketSwitch"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOffSocketSwitch"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Socket Off"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Turn off or unplug devices not in use from the wall
socket."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="8"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.OffLights.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOffLights"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOffLights"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Lights Off"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Turn off the lights when leaving the room."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="9"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.InsulateHome.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToIsolateHome"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToIsolateHome"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Insulate The Home"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Insulate your home and stay warm."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="10"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.Microwave.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="10"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToMicrowaveNotOven"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="10"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToMicrowaveNotOven"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Microwave Don't Oven"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="When possible use the microwave over the oven."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="11"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.ReBatteries.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="11"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReBatteries"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="11"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReBatteries"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Reusable Batteries"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Use Rechargeable Batteries over non rechargeable ones."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="12"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.TurnDownFridge.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="12"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToRefrigiratorDown"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="12"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToRefrigiratorDown"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Turn Down The Fridge"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Turn down the refrigerator to use less energy."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="13"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.InsulateWaterTank.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="13"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToInsulateWater"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="13"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToInsulateWater"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Insulate The Water Tank"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Insulate the water tank and keep the water warm."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="14"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.SealDraft.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="14"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSealDrafts"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="14"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSealDrafts"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Seal The Drafts"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Seal the drafts in your home and keep the cold outside."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="15"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Energy.SealDuct.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="15"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSealDucts"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="15"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSealDucts"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Seal The Ducts"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Seal the ducts in your home and keep the cold out."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The EnergyPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the EnergyPage View Class. This page displays and allows the 
navigation to each of the actions in the Energy category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Energy;
using Application_Green_Quake.Views.EcoActions.Habits;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class EnergyPage : ContentPage
    {
        public EnergyPage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to DryerFull.
       */
        private async void NavigateToDryerFull(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new DryerFull());
        }
        /** This function navigates to MachineFull.
       */
        private async void NavigateToMachineFull(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new MachineFull());
        }
        /** This function navigates to DishwasherFull.
       */
        private async void NavigateToDishwasherFull(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new DishwasherFull());
        }
        /** This function navigates to HangDry.
       */
        private async void NavigateToHangDry(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new HangDry());

        }
        /** This function navigates to EfficientThermostat.
       */
        private async void NavigateToEfficientThermostat(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EfficientThermostat());

        }
        /** This function navigates to SolarPanel.
       */
```

```
        private async void NavigateToSolarPanel(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SolarPanel());
        }
        /** This function navigates to LedLightBulb.
       */
        private async void NavigateToLedLightbulb(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new LedLightBulb());
        }
        /** This function navigates to OffSocketSwitch.
       */
        private async void NavigateToOffSocketSwitch(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new OffSocketSwitch());
        }
        /** This function navigates to TurnOffLights.
       */
        private async void NavigateToOffLights(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new TurnOffLights());
        }
        /** This function navigates to IsolateHome.
       */
        private async void NavigateToIsolateHome(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new IsolateHome());
        }
        /** This function navigates to MicrowaveNotOven.
       */
        private async void NavigateToMicrowaveNotOven(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new MicrowaveNotOven());
        }
        /** This function navigates to ReBatteries.
       */
        private async void NavigateToReBatteries(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ReBatteries());
        }
        /** This function navigates to RefrigiratorDown.
       */
        private async void NavigateToRefrigiratorDown(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new RefrigiratorDown());
        }
        /** This function navigates to InsulateWater.
       */
        private async void NavigateToInsulateWater(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new InsulateWater());
        }
        /** This function navigates to SealDrafts.
       */
        private async void NavigateToSealDrafts(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SealDrafts());
        }
        /** This function navigates to SealDucts.
       */
        private async void NavigateToSealDucts(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SealDucts());
        }
         /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.FoodAndDrinkPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Food and Drink Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Food and Drink" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.OrganicFood.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBuyOrganicFood"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBuyOrganicFood"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Go Organic"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase or make organic food over non organic."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.WaterOverFiz.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWaterOverFizzy"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWaterOverFizzy"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Keeping It H2O"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase or have water over other drinks."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.EatAllYouMake.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEatAllYouMake"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEatAllYouMake"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Eat Everything"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Make just enough food so you can eat it all."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.SaveLeftovers.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSaveLeftovers"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSaveLeftovers"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Save Leftovers"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Save the leftovers for another time."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.NoMeat.jpg}"/>
                <StackLayout Grid.Column="0"
```

```
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToNoMeat"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToNoMeat"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="No Meat"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Eat something else over meat today."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="5"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.ReCoffeeMug.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReCoffeMug"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReCoffeMug"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Use A Reusable Cup"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Use a reusable coffee mug and reduce littering from coffee 
cups."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="6"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.FoodDelivered.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToFoodDelivered"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToFoodDelivered"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Have All Food Delivered"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Instead of traveling to the shops have all your food 
delivered."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="7"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.OwnCoffee.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOwnCoffee"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOwnCoffee"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Brew Your Coffee"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Brew your own coffee instead of buying it."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="8"
                       Aspect="AspectFill"
                       Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.FD.SteeStraw.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSteelStraw"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSteelStraw"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Use Steel Straws"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Buy and use steel straws over plastic or paper ones to reduce waste."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="9"
                       Aspect="AspectFill"
                       Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Use A Reusable Bottle"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use a reusable water bottle instead of using 
plastic bottles."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The FoodAndDrinkPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the FoodAndDrinkPage View Class. This page displays and allows the 
navigation to each of the actions in the FoodAndDrink category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.FoodAndDrink;
using Application_Green_Quake.Views.EcoActions.Water;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FoodAndDrinkPage : ContentPage
    {
        public FoodAndDrinkPage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to BuyOrganicFood.
        */
        private async void NavigateToBuyOrganicFood(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new BuyOrganicFood());
        }
        /** This function navigates to WaterOverFizzy.
        */
        private async void NavigateToWaterOverFizzy(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new WaterOverFizzy());
        }
        /** This function navigates to EatAllYouMake.
        */
        private async void NavigateToEatAllYouMake(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EatAllYouMake());
        }
        /** This function navigates to SaveLeftOvers.
        */
        private async void NavigateToSaveLeftovers(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SaveLeftOvers());
        }
        /** This function navigates to NoMeat.
        */
        private async void NavigateToNoMeat(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new NoMeat());
        }
        /** This function navigates to ReCoffeeMug.
        */
        private async void NavigateToReCoffeMug(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ReCoffeeMug());
        }
        /** This function navigates to FoodDelivered.
        */
        private async void NavigateToFoodDelivered(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new FoodDelivered());
        }
        /** This function navigates to OwnCoffee.
        */
        private async void NavigateToOwnCoffee(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new OwnCoffee());
        }
        /** This function navigates to SteelStraw.
        */
        private async void NavigateToSteelStraw(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SteelStraw());
        }
        /** This function navigates to ReusableWater.
        */
        private async void NavigateToReusableWater(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ReusableWater());
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.HabitsPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Habits Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Habits" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.TimedBrushing.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBrushingPage"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBrushingPage"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Tap Off"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Turn off the tap when you are brushing your teeth and save
```

```
water and the planet."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.TimedShower.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToTimedShowerPage"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToTimedShowerPage"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Time Your Shower"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Time your showers so they don't take too long and save
water."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.ShowerNoBath.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToShoweredInsteadOfBath"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToShoweredInsteadOfBath"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Shower No Bath"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Take a brief shower over taking a bath."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.FullDishwasher.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Full Dishwasher"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Only use the dishwasher when it is full."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.OffLights.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
```

```
                        <TapGestureRecognizer Tapped="NavigateToTurnOffLights"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToTurnOffLights"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Turn Off The Lights"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Turn off the lights when leaving the room and save 
energy."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="5"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.MatchOverLighter.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToUseMatches"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToUseMatches"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Matches Over Lighters"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Use Matches over lighters as they are more environmentally 
friendly."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The HabitsPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the HabitsPage View Class. This page displays and allows the
navigation to each of the actions in the Habits category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Habits;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class HabitsPage : ContentPage
    {
        public HabitsPage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to BrushingTeeth.
        */
        private async void NavigateToBrushingPage(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new BrushingTeeth());
        }
        /** This function navigates to TimedShower.
        */
        private async void NavigateToTimedShowerPage(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new TimedShower());
        }
        /** This function navigates to ShowerInstead.
        */
        private async void NavigateToShoweredInsteadOfBath(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ShowerInstead());
        }
        /** This function navigates to DishwasherFull.
        */
        private async void NavigateToDishwasherFull(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new DishwasherFull());

        }
        /** This function navigates to TurnOffLights.
        */
        private async void NavigateToTurnOffLights(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new TurnOffLights());

        }
        /** This function navigates to UseMatches.
        */
        private async void NavigateToUseMatches(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new UseMatches());
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.HomePage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Home Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Home" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" 
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.AirOut.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToAirOutHome"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToAirOutHome"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Air Out Your Home"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Air out your home. Let some fresh air inside."
```

```
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Home.Outside.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOutsideOnce"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOutsideOnce"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Go Outside"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="When brushing your teeth it can be easy to leave the water running but this attributes to major water waste."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Home.PlantHome.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPlantIntoHome"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
```

```
                        <TapGestureRecognizer Tapped="NavigateToPlantIntoHome"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Bring A Plant Inside"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Bring a plant into your home."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.NonHarmful.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToNonHarmfulProducts"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToNonHarmfulProducts"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Use Non Harmful Products"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"
                           HorizontalTextAlignment="Center"/>
                    <Label Text="Use non harmful bio products instead of harmful ones when 
possible."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.Flush.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSaveFlush"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSaveFlush"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Save A Flush"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Save a flush when you can and save water."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="5"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.Napkin.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToClothNapkins"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToClothNapkins"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Cloth Napkins"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Use Cloth Napkins over paper ones."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="6"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.Towel.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToClothTowels"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToClothTowels"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Cloth Towels"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Use Cloth Towels over paper ones."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The HomePage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the HomePage View Class. This page displays and allows the navigation to each of the actions in the Home category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Home;
using Application_Green_Quake.Views.EcoActions.Shopping;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class HomePage : ContentPage
    {
        public HomePage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to AirOutHome.
        */
        private async void NavigateToAirOutHome(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new AirOutHome());

        }
        /** This function navigates to OutsideOnce.
        */
        private async void NavigateToOutsideOnce(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new OutsideOnce());

        }
        /** This function navigates to PlantIntoHome.
        */
        private async void NavigateToPlantIntoHome(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PlantIntoHome());

        }
        /** This function navigates to NonHarmfulProducts.
        */
        private async void NavigateToNonHarmfulProducts(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new NonHarmfulProducts());

        }
        /** This function navigates to ToiletFlushes.
        */
        private async void NavigateToSaveFlush(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ToiletFlushes());

        }
        /** This function navigates to ClothNapkins.
        */
        private async void NavigateToClothNapkins(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ClothNapkins());

        }
        /** This function navigates to ClothTowels.
        */
        private async void NavigateToClothTowels(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ClothTowels());

        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.OutdoorsPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Outdoors Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Outdoors" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource Application_Green_Quake.Images.SubCategories.Outdoors.Tree.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPlantATree"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPlantATree"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Plant A Tree"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Plant a tree somewhere you can and watch it grow."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.Flower.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPlantAFlower"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPlantAFlower"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Plant A Flower"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Plant a flower in your garden. It doesn't just look good 
it improves the environment"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.Bush.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPlantABush"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
```

```
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPlantABush"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Plant A Bush"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Plant a bush and let it grow."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.Picnic.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPicnic"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPicnic"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Have A Picnic"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Go outside and Have a Picnic."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.Camping.jpg}"/>
                <StackLayout Grid.Column="0"
```

```
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToGoCamping"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToGoCamping"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Go Camping"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Go Camping and have some fun. Use reusable gear."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="5"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.Scoop.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToScoop"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToScoop"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Scoop da Poop"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="When your dog makes a mess clean it up. It's only right."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="6"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.HerbGarden.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpHerbGarden"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpHerbGarden"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Set Up A Herb Garden"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Set up a herb garden and plant some herbs."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="7"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.VegGarden.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpVegetableGarden"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpVegetableGarden"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Set Up A Vegetable Garden"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"
                           HorizontalTextAlignment="Center"/>
                    <Label Text="Set up a Vegetable garden and plant some vegetable."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="8"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.FruitGarden.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpFruitGarden"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpFruitGarden"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Set Up A Fruit Garden"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Set up a Fruit garden and plant some fruit."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="9"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Water.RainBarrel.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpRainWaterColector"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
```

```
                             Grid.Row="9"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpRainWaterColector"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Set Up A Rain Barrel"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Set up a rain barrel and collect rain water that you can 
use."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="10"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Outdoors.BirdFeeder.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="10"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpBirdfeeder"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="10"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpBirdfeeder"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Set Up A Bird Feeder"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Set up a bird feeder in your garden and help out the 
wildlife."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
```

```
</ContentPage>
```

```
/*! \class The OutdoorsPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the OutdoorsPage View Class. This page displays and allows the 
navigation to each of the actions in the Outdoors category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Outdoors;
using Application_Green_Quake.Views.EcoActions.Water;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class OutdoorsPage : ContentPage
    {
        public OutdoorsPage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to PlantATree.
        */
        private async void NavigateToPlantATree(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PlantATree());
        }
        /** This function navigates to PlantAFlower.
        */
        private async void NavigateToPlantAFlower(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PlantAFlower());
        }
        /** This function navigates to PlantABush.
        */
        private async void NavigateToPlantABush(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PlantABush());
        }
        /** This function navigates to Picnic.
        */
        private async void NavigateToPicnic(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Picnic());
        }
        /** This function navigates to GoCamping.
        */
        private async void NavigateToGoCamping(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new GoCamping());
        }
        /** This function navigates to Scoop.
        */
        private async void NavigateToScoop(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Scoop());
        }
        /** This function navigates to SetUpHerbGarden.
        */
        private async void NavigateToSetUpHerbGarden(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SetUpHerbGarden());
        }
        /** This function navigates to SetUpVegetableGarden.
        */
        private async void NavigateToSetUpVegetableGarden(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SetUpVegetableGarden());
        }
        /** This function navigates to SetUpFruitGarden.
        */
        private async void NavigateToSetUpFruitGarden(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SetUpFruitGarden());
        }
        /** This function navigates to RainBarrel.
        */
        private async void NavigateToSetUpRainWaterColector(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new RainBarrel());
        }
        /** This function navigates to UpBirdfeeder.
        */
        private async void NavigateToSetUpBirdfeeder(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new UpBirdfeeder());
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.ShoppingPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Shopping Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Shopping" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" 
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Use A Reusable Bottle"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use a reusable water bottle instead of using 
plastic bottles."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.ReBag.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPurchaseReusableBag"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPurchaseReusableBag"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Reusable Bag"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use a reusable bag. Don't use bags that you 
throw away."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.LocalProduct.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToLocalProduct"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToLocalProduct"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Buy Local"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Buy Local products and support the community."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.OrganicFood.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOrganicFood"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOrganicFood"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Go Organic"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and make organic food over non organic"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
```

```
                           Aspect="AspectFill"
                           Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.FoodBulk.jpg}"/>
                    <StackLayout Grid.Column="0"
                                 Grid.Row="4"
                                 BackgroundColor="Black"
                                 Opacity=".5">
                        <StackLayout.GestureRecognizers>
                            <TapGestureRecognizer Tapped="NavigateToFoodInBulk"/>
                        </StackLayout.GestureRecognizers>
                    </StackLayout>
                    <StackLayout Grid.Column="0"
                                 Grid.Row="4"
                                 VerticalOptions="End"
                                 Spacing="5"
                                 Margin="40,0,40,15">
                        <StackLayout.GestureRecognizers>
                            <TapGestureRecognizer Tapped="NavigateToFoodInBulk"/>
                        </StackLayout.GestureRecognizers>
                        <Label Text="Buy In Bulk"
                               TextColor="White"
                               HorizontalOptions="Center"
                               FontSize="24"
                               FontAttributes="Bold"
                               FontFamily="Proxima Nova"/>
                        <Label Text="Purchase food in bulk to save needless trips to the 
shops."
                               TextColor="White"
                               HorizontalOptions="Center"
                               FontSize="15"
                               FontFamily="Proxima Nova Thin"
                               HorizontalTextAlignment="Center"/>
                    </StackLayout>

                    <Image Grid.Column="0"
                           Grid.Row="5"
                           Aspect="AspectFill"
                           Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.EcoProduct.jpg}"/>
                    <StackLayout Grid.Column="0"
                                 Grid.Row="5"
                                 BackgroundColor="Black"
                                 Opacity=".5">
                        <StackLayout.GestureRecognizers>
                            <TapGestureRecognizer Tapped="NavigateToEcoFriendlyProduct"/>
                        </StackLayout.GestureRecognizers>
                    </StackLayout>
                    <StackLayout Grid.Column="0"
                                 Grid.Row="5"
                                 VerticalOptions="End"
                                 Spacing="5"
                                 Margin="40,0,40,15">
                        <StackLayout.GestureRecognizers>
                            <TapGestureRecognizer Tapped="NavigateToEcoFriendlyProduct"/>
                        </StackLayout.GestureRecognizers>
                        <Label Text="Eco Product"
                               TextColor="White"
                               HorizontalOptions="Center"
                               FontSize="24"
                               FontAttributes="Bold"
                               FontFamily="Proxima Nova"/>
                        <Label Text="Purchase and Eco Friendly Product over a non Eco Friendly 
```

```
Product."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="6"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.EthicalClothes.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEthicalClothes"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEthicalClothes"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Purchase Ethical Clothes"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Instead of purchasing clothes that are not ethical 
purchase Ethical Clothes."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="7"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.EcoBrush.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoFriendlyToothbrush"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoFriendlyToothbrush"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Eco Toothbrush"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use an Eco Friendly Toothbrush and an Eco 
Friendly Toothbrush."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="8"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.EcoAppliance.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoFreidnlyApplicance"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoFreidnlyApplicance"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Eco Appliance"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use and Eco Friendly Appliance."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="9"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.LooseTea.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToLooseLeafTea"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToLooseLeafTea"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Loose Leaf Tea"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="No need for bagged tea. Just use Loose Leaf Tea instead."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="10"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.ReBatteries.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="10"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReBatteries"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="10"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReBatteries"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Reusable Batteries"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use Rechargeable Batteries over non
rechargeable ones."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="11"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.Napkin.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="11"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToClothNapkins"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="11"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToClothNapkins"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Cloth Napkins"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use Cloth Napkins over paper ones."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="12"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Shopping.Towel.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="12"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToClothTowels"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="12"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToClothTowels"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Cloth Towels"
                           TextColor="White"
                           HorizontalOptions="Center"
```

```
                                   FontSize="24"
                                   FontAttributes="Bold"
                                   FontFamily="Proxima Nova"/>
                            <Label Text="Purchase and use Cloth Towels over paper ones."
                                   TextColor="White"
                                   HorizontalOptions="Center"
                                   FontSize="15"
                                   FontFamily="Proxima Nova Thin"
                                   HorizontalTextAlignment="Center"/>
                        </StackLayout>
                    </Grid>
                </ScrollView>
            </ContentPage.Content>
</ContentPage>
```

```
/*! \class The ShoppingPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the ShoppingPage View Class. This page displays and allows the
navigation to each of the actions in the Shopping category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Shopping;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ShoppingPage : ContentPage
    {
        public ShoppingPage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to PurchaseReusableWater.
        */
        private async void NavigateToReusableWater(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PurchaseReusableWater());
        }
        /** This function navigates to ReusableBag.
        */
        private async void NavigateToPurchaseReusableBag(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ReusableBag());
        }
        /** This function navigates to LocalProduct.
        */
        private async void NavigateToLocalProduct(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new LocalProduct());
        }
        /** This function navigates to OrganicFood.
        */
        private async void NavigateToOrganicFood(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new OrganicFood());
        }
        /** This function navigates to FoodInBulk.
        */
        private async void NavigateToFoodInBulk(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new FoodInBulk());
        }
        /** This function navigates to EcoFriendlyProduct.
        */
        private async void NavigateToEcoFriendlyProduct(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EcoFriendlyProduct());
        }
        /** This function navigates to EthicalClothes.
        */
        private async void NavigateToEthicalClothes(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EthicalClothes());
        }
        /** This function navigates to EcoFriendlyToothbrush.
        */
        private async void NavigateToEcoFriendlyToothbrush(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EcoFriendlyToothbrush());
        }
        /** This function navigates to EcoFreidnlyApplicance.
        */
        private async void NavigateToEcoFreidnlyApplicance(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EcoFreidnlyApplicance());
        }
        /** This function navigates to LooseLeafTea.
        */
        private async void NavigateToLooseLeafTea(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new LooseLeafTea());
        }
        /** This function navigates to ReBatteries.
        */
        private async void NavigateToReBatteries(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ReBatteries());
        }
        /** This function navigates to ClothNapkins.
        */
        private async void NavigateToClothNapkins(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ClothNapkins());
        }
        /** This function navigates to ClothTowels.
        */
        private async void NavigateToClothTowels(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ClothTowels());
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.TravelPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Travel Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Travel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Carpool.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCarpool"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCarpool"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Carpool"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Carpool when traveling to work or school."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.PublicTransport.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Public Transport"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Use Public Transport over a car."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.Walk.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWalk"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWalk"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Walk"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Walk there if you can."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Cycle.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCycle"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCycle"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Cycle"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Cycle instead if you can."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.EcoCar.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
```

```
                        <TapGestureRecognizer Tapped="NavigateToEcoFreindlyCar"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoFreindlyCar"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Eco Car"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use and Eco Friendly Car."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The TravelPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the TravelPage View Class. This page displays and allows the navigation to each of the actions in the Travel category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Travel;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TravelPage : ContentPage
    {
        public TravelPage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to Carpool.
        */
        private async void NavigateToCarpool(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Carpool());
        }
        /** This function navigates to PublicTransport.
        */
        private async void NavigateToPublicTransport(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PublicTransport());
        }
        /** This function navigates to Walk.
        */
        private async void NavigateToWalk(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Walk());
        }
        /** This function navigates to Cycle.
        */
        private async void NavigateToCycle(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Cycle());
        }
        /** This function navigates to EcoFreindlyCar.
        */
        private async void NavigateToEcoFreindlyCar(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EcoFreindlyCar());
        }
        /** This function is called before the page is displayed and it created an object ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.WastePage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Waste Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Waste" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" 
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.Recycled.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToUseRecyclingBin"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToUseRecyclingBin"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Recycle"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Recycle your waste."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.SetUpBins.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpRecyclingBin"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpRecyclingBin"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Set Up Recycling Bins"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Set up recycling bins so you can begin you recycling 
journey."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.Compost.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpCompostBin"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToSetUpCompostBin"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Set Up Compost Bins"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Set up compost bins so you can start composting you food 
waste."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.BioBags.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToUseBiogradableBinBags"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToUseBiogradableBinBags"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Bio Bin Bags"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use Biodegradable Bin Bags instead."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Waste.OnlineBills.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
```

```
                        <TapGestureRecognizer Tapped="NavigateToPayBillsOnline"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPayBillsOnline"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Pay Bills Online"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Pay your Bills online. No need to waste paper.."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The WastePage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WastePage View Class. This page displays and allows the 
navigation to each of the actions in the Waste category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Waste;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WastePage : ContentPage
    {
        public WastePage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to UseRecyclingBin.
        */
        private async void NavigateToUseRecyclingBin(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new UseRecyclingBin());
        }
        /** This function navigates to SetUpRecyclingBin.
        */
        private async void NavigateToSetUpRecyclingBin(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SetUpRecyclingBin());
        }
        /** This function navigates to CompostWaste.
        */
        private async void NavigateToSetUpCompostBin(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new CompostWaste());
        }
        /** This function navigates to UseBiogradableBinBags.
        */
        private async void NavigateToUseBiogradableBinBags(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new UseBiogradableBinBags());
        }
        /** This function navigates to BillsOnline.
        */
        private async void NavigateToPayBillsOnline(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new BillsOnline());
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.WaterPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Water Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Water" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" 
Margin="0,0,30,0"/>
        </StackLayout>

    </NavigationPage.TitleView>
    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.TimedBrushing.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBrushingPage"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBrushingPage"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Tap Off"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Turn off the tap when you are brushing your teeth and save 
water and the planet."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.TimedShower.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToTimedShowerPage"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToTimedShowerPage"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Time Your Shower"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Time your showers so they don't take too long and save 
water."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.ShowerNoBath.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToShoweredInsteadOfBath"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToShoweredInsteadOfBath"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Shower No Bath"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Take a brief shower over taking a bath."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Habits.FullDishwasher.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Full Dishwasher"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Only use the dishwasher when it is full."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Use A Reusable Bottle"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use a reusable water bottle instead of using 
plastic bottles."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="5"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Water.ShowerHead.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWSShowerhead"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWSShowerhead"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Water Saving Shower Head"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Install a Water Saving Shower Head to use less water."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="6"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Water.CisternDis.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCisternDisplacement"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCisternDisplacement"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Cistern Displacement System"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Install a Cistern Displacement System to save more water."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="7"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Water.ShowerBucket.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToShowerBucket"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToShowerBucket"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Shower Bucket"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Collect the water when you shower and use it for something 
else."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="8"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Water.RainBarrel.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToRainBarrel"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToRainBarrel"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Rain Barrel"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Set up a Rain Barrel to collect rainwater which can be 
reused for something else.."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="9"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Home.Flush.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToToiletFlushes"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="9"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToToiletFlushes"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Save A Flush"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Save a flush when you can and save water."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```csharp
/*! \class The WaterPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WaterPage View Class. This page displays and allows the navigation to each of the actions in the Water category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Habits;
using Application_Green_Quake.Views.EcoActions.Home;
using Application_Green_Quake.Views.EcoActions.Water;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WaterPage : ContentPage
    {
        public WaterPage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to BrushingTeeth.
        */
        private async void NavigateToBrushingPage(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new BrushingTeeth());
        }
        /** This function navigates to TimedShower.
        */
        private async void NavigateToTimedShowerPage(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new TimedShower());
        }
        /** This function navigates to ShowerInstead.
        */
        private async void NavigateToShoweredInsteadOfBath(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ShowerInstead());
        }
        /** This function navigates to DishwasherFull.
        */
        private async void NavigateToDishwasherFull(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new DishwasherFull());

        }
        /** This function navigates to ReusableWater.
        */
        private async void NavigateToReusableWater(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ReusableWater());
        }
        /** This function navigates to WSShowerHead.
        */
        private async void NavigateToWSShowerhead(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new WSShowerHead());
        }
        /** This function navigates to CisternDisplacement.
        */
        private async void NavigateToCisternDisplacement(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new CisternDisplacement());
        }
        /** This function navigates to ShowerBucket.
        */
        private async void NavigateToShowerBucket(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ShowerBucket());
        }
        /** This function navigates to RainBarrel.
        */
        private async void NavigateToRainBarrel(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new RainBarrel());
        }
        /** This function navigates to ToiletFlushes.
        */
        private async void NavigateToToiletFlushes(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ToiletFlushes());
        }
        /** This function is called before the page is displayed and it created an object 
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.WorkPage"
             xmlns:local="clr-namespace:Application_Green_Quake.Models"
             Title="Work Subcategories">

    <NavigationPage.TitleView>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Fill" 
VerticalOptions="EndAndExpand" Spacing="0">
            <Label Text="Work" TextColor="White" FontSize="20" FontAttributes="Italic" 
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
            <Label x:Name="theLevel" TextColor="White" FontSize="20" 
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" 
Margin="0,0,30,0"/>
        </StackLayout>
    </NavigationPage.TitleView>

    <ContentPage.Content>
        <ScrollView>
            <Grid Margin="5,5,5,5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Image Grid.Column="0"
                       Grid.Row="0"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.Carpool.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCarpool"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="0"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCarpool"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Carpool"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Carpool when traveling to work or school."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="1"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.PublicTransport.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="1"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Public Transport"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Use Public Transport over a car."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="2"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Cycle.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCycle"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="2"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToCycle"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Cycle"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Cycle instead if you can."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="3"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.Walk.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWalk"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="3"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWalk"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Walk"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Walk there if you can."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="4"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Travel.EcoCar.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoFreindlyCar"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="4"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToEcoFreindlyCar"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Eco Cart"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Purchase and use and Eco Friendly Car."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="5"
                       Aspect="AspectFill"
                       Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.OffLights.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOffLights"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="5"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOffLights"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Lights Off"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Turn off the lights when leaving the room."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="6"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Work.Off.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOffElectronics"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="6"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToOffElectronics"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Electronics Off"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Switch off the Electronics that are not in use."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="7"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Work.Remote.jpg}"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             BackgroundColor="Black"
                             Opacity=".5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWorkingRemotely"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="7"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToWorkingRemotely"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Remote Work"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Work Remotely if you can."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>

                <Image Grid.Column="0"
                       Grid.Row="8"
                       Aspect="AspectFill"
                       Source="{local:ImageResource 
Application_Green_Quake.Images.SubCategories.Work.Paper.jpg}"
                       Margin="0,0,0,5"/>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             BackgroundColor="Black"
                             Opacity=".5"
                             Margin="0,0,0,5">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBothSidesPaper"/>
                    </StackLayout.GestureRecognizers>
                </StackLayout>
                <StackLayout Grid.Column="0"
                             Grid.Row="8"
                             VerticalOptions="End"
                             Spacing="5"
                             Margin="40,0,40,15">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBothSidesPaper"/>
                    </StackLayout.GestureRecognizers>
                    <Label Text="Both Sides"
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="24"
                           FontAttributes="Bold"
                           FontFamily="Proxima Nova"/>
                    <Label Text="Use Both Sides of the paper when you are using it."
                           TextColor="White"
                           HorizontalOptions="Center"
                           FontSize="15"
                           FontFamily="Proxima Nova Thin"
                           HorizontalTextAlignment="Center"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The WorkPage View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the WorkPage View Class. This page displays and allows the
navigation to each of the actions in the Work category.
 *
 */
using Application_Green_Quake.ViewModels;
using Application_Green_Quake.Views.EcoActions.Habits;
using Application_Green_Quake.Views.EcoActions.Travel;
using Application_Green_Quake.Views.EcoActions.Work;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WorkPage : ContentPage
    {
        public WorkPage()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function navigates to Carpool.
        */
        private async void NavigateToCarpool(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Carpool());
        }
        /** This function navigates to PublicTransport.
        */
        private async void NavigateToPublicTransport(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PublicTransport());
        }
        /** This function navigates to Cycle.
        */
        private async void NavigateToCycle(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Cycle());
        }
        /** This function navigates to Walk.
        */
        private async void NavigateToWalk(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new Walk());
        }
        /** This function navigates to EcoFreindlyCar.
        */
        private async void NavigateToEcoFreindlyCar(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new EcoFreindlyCar());
        }
        /** This function navigates to TurnOffLights.
        */
        private async void NavigateToOffLights(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new TurnOffLights());
        }
        /** This function navigates to OffElectronics.
        */
        private async void NavigateToOffElectronics(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new OffElectronics());
        }
        /** This function navigates to WorkingRemotely.
        */
        private async void NavigateToWorkingRemotely(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new WorkingRemotely());
        }
        /** This function navigates to BothSidesPaper.
        */
        private async void NavigateToBothSidesPaper(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new BothSidesPaper());
        }
        /** This function is called before the page is displayed and it created an object
ans uses it's SetLvl method to set the players level in the app
         * and display it in the navigation bar.
        */
        protected override void OnAppearing()
        {
            GetData data = new GetData();
            data.SetLvl();

            theLevel.Text = "LVL: " + GetData.lvl.ToString();
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.ProfilePage.Achievements"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">
    <ContentPage.Content>
        <ScrollView>
            <Grid BackgroundColor="#002a1e" RowSpacing="0" Padding="10,10,10,10">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>
                <Grid.RowDefinitions>
                    <RowDefinition Height="3200"/>
                </Grid.RowDefinitions>

                <StackLayout Grid.Column="0" Grid.Row="0" VerticalOptions="Start">
                    <Frame CornerRadius="60" HorizontalOptions="Start" Margin="0" 
Padding="0" BackgroundColor="#33554b" >
                        <Image x:Name="a1" Source="{local:ImageResource 
Application_Green_Quake.Images.lockTwo.png}" Aspect="AspectFill"/>
                    </Frame>

                    <Label x:Name="a1Txt" Text="Locked " TextColor="White" 
HorizontalTextAlignment="Center" HeightRequest="60"/>

                    <Frame CornerRadius="60" HorizontalOptions="Start" 
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a5" Source="{local:ImageResource 
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a5Txt" Text="Locked " TextColor="White" 
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" 
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a9" Source="{local:ImageResource 
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a9Txt" Text="Locked " TextColor="White" 
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" 
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a13" Source="{local:ImageResource 
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a13Txt" Text="Locked " TextColor="White" 
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" 
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a17" Source="{local:ImageResource 
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a17Txt" Text="Locked " TextColor="White" 
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" 
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a21" Source="{local:ImageResource 
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
```

```
                    </Frame>

                    <Label x:Name="a21Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a25" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a25Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>

                    <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a29" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a29Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a33" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a33Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a37" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a37Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a41" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a41Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a45" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a45Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a49" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a49Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a53" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
```

```xml
                    </Frame>

                    <Label x:Name="a53Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a57" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a57Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a61" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a61Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a65" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a65Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a69" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a69Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a73" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a73Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a77" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a77Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a81" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a81Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                </StackLayout>

                <StackLayout Grid.Column="1" Grid.Row="0" VerticalOptions="Start">
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b" >
```

```xml
                        <Image x:Name="a2" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a2Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a6" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a6Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a10" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a10Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b" >
                        <Image x:Name="a14" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a14Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a18" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a18Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a22" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a22Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a26" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a26Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a30" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a30Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a34" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a34Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a38" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a38Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a42" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a42Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a46" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a46Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a50" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a50Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a54" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a54Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a58" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a58Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a62" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a62Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a66" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a66Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a70" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a70Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a74" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a74Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a78" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a78Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                </StackLayout>

                <StackLayout Grid.Column="2" Grid.Row="0" VerticalOptions="Start">
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a3" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a3Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a7" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a7Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a11" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a11Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a15" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a15Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a19" Source="{local:ImageResource
```

```
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a19Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a23" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a23Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a27" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a27Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a31" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a31Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a35" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a35Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a39" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a39Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a43" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a43Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a47" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a47Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a51" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
```

```xml
                    </Frame>

                    <Label x:Name="a51Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a55" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a55Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a59" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a59Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a63" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a63Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a67" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a67Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a71" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a71Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a75" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a75Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a79" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a79Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                </StackLayout>

                <StackLayout Grid.Column="3" Grid.Row="0" VerticalOptions="Start">
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
```

```xml
                        <Image x:Name="a4" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a4Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a8" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a8Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a12" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a12Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a16" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a16Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a20" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a20Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a24" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a24Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a28" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a28Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a32" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a32Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a36" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a36Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a40" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a40Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a44" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a44Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a48" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a48Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a52" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a52Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a56" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a56Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a60" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a60Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a64" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a64Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a68" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>
```

```xml

                    <Label x:Name="a68Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a72" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a72Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a76" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a76Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a80" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a80Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center" HeightRequest="60"/>
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The Achievements View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Achievements View Class. This class is the class that displays a
Achievements on the Achievements page.
 *
 */
using Application_Green_Quake.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.ProfilePage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Achievements : ContentPage
    {
        IAuth auth;

        public Achievements()
        {
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();
            OnAppearing();
        }
        /** This function is called before the page is displayed. It displays the images as
are met
        */
        protected override void OnAppearing()
        {
            if (GetAchievementsData.brushingCount >= 5 && GetAchievementsData.brushingCount
            {
                a1.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingBronze.
                a1Txt.Text = "Brushing Bronze";
            }
            else if (GetAchievementsData.brushingCount >= 15 && GetAchievementsData.brushing
            {
                a1.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingSilver.
                a1Txt.Text = "Brushing Silver";
            }
            else if (GetAchievementsData.brushingCount >= 25)
            {
                a1.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingGold.pn
                a1Txt.Text = "Brushing Gold";
            }

            if (GetAchievementsData.fullWasherCount >= 5 && GetAchievementsData.fullWasherCo
            {
                a2.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterBronz
                a2Txt.Text = "Dish Washer Bronze";
            }
            else if (GetAchievementsData.fullWasherCount >= 15 && GetAchievementsData.fullWa
            {
                a2.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterSilve
                a2Txt.Text = "Dish Washer Silver";
            }
            else if (GetAchievementsData.fullWasherCount >= 25)
            {
                a2.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterGold.
                a2Txt.Text = "Dish Washer Gold";
            }

            if (GetAchievementsData.shareCount >= 5 && GetAchievementsData.shareCount < 15)
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerBronze.pn
                a3Txt.Text = "Shower Bronze";
            }
            else if (GetAchievementsData.shareCount >= 15 && GetAchievementsData.shareCount
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerSilver.pn
                a3Txt.Text = "Shower Silver";
            }
            else if (GetAchievementsData.shareCount >= 25)
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerGold.png"
                a3Txt.Text = "Shower Gold";
            }

            if (GetAchievementsData.timedShowerCount >= 5 && GetAchievementsData.timedShower
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerBronz
                a4Txt.Text = "Timed Shower Bronze";
            }
            else if (GetAchievementsData.timedShowerCount >= 15 && GetAchievementsData.timed
25)
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerSilve
                a4Txt.Text = "Timed Shower Silver";
            }
            else if (GetAchievementsData.timedShowerCount >= 25)
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerGold.
                a4Txt.Text = "Timed Shower Gold";
            }

            if (GetAchievementsData.offLigtsCount >= 5 && GetAchievementsData.offLigtsCount
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightBronze.png
                a5Txt.Text = "Off Lights Bronze";
            }
            else if (GetAchievementsData.offLigtsCount >= 15 && GetAchievementsData.offLigts
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightSilver.png
                a5Txt.Text = "Off Lights Silver";
            }
            else if (GetAchievementsData.offLigtsCount >= 25)
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightGold.png")
```

```
                a5Txt.Text = "Off Lights Gold";
            }

            if (GetAchievementsData.matchesCount >= 5 && GetAchievementsData.matchesCount <
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.matchBronze.png
                a6Txt.Text = "Matches Bronze";
            }
            else if (GetAchievementsData.matchesCount >= 15 && GetAchievementsData.matchesCo
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.matchSilver.png
                a6Txt.Text = "Matches Silver";
            }
            else if (GetAchievementsData.matchesCount >= 25)
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.matchGold.png")
                a6Txt.Text = "Matches Gold";
            }

            if (GetAchievementsData.foodDeliverCount >= 5 && GetAchievementsData.foodDeliver
            {
                a7.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.deliverBronze.png")
                a7Txt.Text = "Food Delivered Bronze";
            }
            else if (GetAchievementsData.foodDeliverCount >= 15 && GetAchievementsData.foodD
25)
            {
                a7.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.deliverSilver.png")
                a7Txt.Text = "Food Delivered Silver";
            }
            else if (GetAchievementsData.foodDeliverCount >= 25)
            {
                a7.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.deliverGold.png");
                a7Txt.Text = "Food Delivered Gold";
            }

            if (GetAchievementsData.eatAllCount >= 5 && GetAchievementsData.eatAllCount < 15
            {
                a8.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.eatAllBronze.png");
                a8Txt.Text = "Eat All Bronze";
            }
            else if (GetAchievementsData.eatAllCount >= 15 && GetAchievementsData.eatAllCoun
            {
                a8.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.eatAllSilver.png");
                a8Txt.Text = "Eat All Silver";
            }
            else if (GetAchievementsData.eatAllCount >= 25)
            {
                a8.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.eatAllGold.png");
                a8Txt.Text = "Eat All Gold";
            }

            if (GetAchievementsData.saveLeftOversCount >= 5 && GetAchievementsData.saveLeftO
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.leftoversBronze.png
                a9Txt.Text = "Save Leftovers Bronze";
            }
            else if (GetAchievementsData.saveLeftOversCount >= 15 && GetAchievementsData.sav
< 25)
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.leftoversSilver.png
                a9Txt.Text = "Save Leftovers Silver";
            }
            else if (GetAchievementsData.saveLeftOversCount >= 25)
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.leftoversGold.png")
                a9Txt.Text = "Save Leftovers Gold";
            }

            if (GetAchievementsData.noMeatCount >= 5 && GetAchievementsData.noMeatCount < 15
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.noMeatBronze.png");
                a10Txt.Text = "No Meat Bronze";
            }
            else if (GetAchievementsData.noMeatCount >= 15 && GetAchievementsData.noMeatCoun
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.noMeatSilver.png");
                a10Txt.Text = "No Meat Silver";
            }
            else if (GetAchievementsData.noMeatCount >= 25)
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.noMeatGold.png");
                a10Txt.Text = "No Meat Gold";
            }

            if (GetAchievementsData.organicFoodCount >= 5 && GetAchievementsData.organicFood
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.organicBronze.png")
                a11Txt.Text = "Organic Food Bronze";
            }
            else if (GetAchievementsData.organicFoodCount >= 15 && GetAchievementsData.organ
25)
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.organicSilver.png")
                a11Txt.Text = "Organic Food Silver";
            }
            else if (GetAchievementsData.organicFoodCount >= 25)
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.organicGold.png");
                a11Txt.Text = "Organic Food Gold";
            }

            if (GetAchievementsData.ownCoffeeCount >= 5 && GetAchievementsData.ownCoffeeCoun
            {
                a12.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.ownCoffeeBronze.png
                a12Txt.Text = "Own Coffee Bronze";
            }
            else if (GetAchievementsData.ownCoffeeCount >= 15 && GetAchievementsData.ownCoff
            {
                a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.ownCoffeeSilver.png
                a12Txt.Text = "Own Coffee Silver";
            }
            else if (GetAchievementsData.ownCoffeeCount >= 25)
            {
                a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.ownCoffeeGold.png")
                a12Txt.Text = "Own Coffee Gold";
            }

            if (GetAchievementsData.reCoffeeMugCount >= 5 && GetAchievementsData.reCoffeeMug
            {
                a13.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.reCupBronze.png");
                a13Txt.Text = "Reusable Mug Bronze";
            }
            else if (GetAchievementsData.reCoffeeMugCount >= 15 && GetAchievementsData.reCof
25)
            {
                a13.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.reCupSilver.png");
                a13Txt.Text = "Reusable Mug Silver";
            }
            else if (GetAchievementsData.reCoffeeMugCount >= 25)
            {
                a13.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.reCupGold.png");
                a13Txt.Text = "Reusable Mug Gold";
            }

            if (GetAchievementsData.steelStrawCount >= 5 && GetAchievementsData.steelStrawCo
            {
                a14.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawBronze.png");
                a14Txt.Text = "Steel Straw Bronze";
            }
            else if (GetAchievementsData.steelStrawCount >= 15 && GetAchievementsData.steelS
            {
                a14.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawSilver.png");
                a14Txt.Text = "Steel Straw Silver";
            }
            else if (GetAchievementsData.steelStrawCount >= 25)
            {
                a14.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawGold.png");
                a14Txt.Text = "Steel Straw Gold";
            }

            if (GetAchievementsData.waterOverFizzyCount >= 5 && GetAchievementsData.waterOve
15)
            {
                a15.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterBronze.png");
                a15Txt.Text = "Water Bronze";
            }
            else if (GetAchievementsData.waterOverFizzyCount >= 15 &&
GetAchievementsData.waterOverFizzyCount < 25)
            {
                a15.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterSilver.png");
                a15Txt.Text = "Water Silver";
            }
            else if (GetAchievementsData.waterOverFizzyCount >= 25)
            {
                a15.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterGold.png");
                a15Txt.Text = "Water Gold";
            }

            if (GetAchievementsData.fullDryerCount >= 5 && GetAchievementsData.fullDryerCoun
            {
                a16.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerBronze.png
                a16Txt.Text = "Dryer Bronze";
            }
            else if (GetAchievementsData.fullDryerCount >= 15 && GetAchievementsData.fullDry
            {
                a16.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerSilver.png
                a16Txt.Text = "Dryer Silver";
            }
            else if (GetAchievementsData.fullDryerCount >= 25)
            {
                a16.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerGold.png")
                a16Txt.Text = "Dryer Gold";
            }

            if (GetAchievementsData.hangDryCount >= 5 && GetAchievementsData.hangDryCount <
            {
                a17.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangBronze.png"
                a17Txt.Text = "Hang Dry Bronze";
            }
            else if (GetAchievementsData.hangDryCount >= 15 && GetAchievementsData.hangDryCo
            {
                a17.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangSilver.png"
                a17Txt.Text = "Hang Dry Silver";
            }
            else if (GetAchievementsData.hangDryCount >= 25)
            {
                a17.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangGold.png");
                a17Txt.Text = "Hang Dry Gold";
            }

            if (GetAchievementsData.microwaveCount >= 5 && GetAchievementsData.microwaveCoun
            {
                a18.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveBronze
                a18Txt.Text = "Microwave Bronze";
            }
            else if (GetAchievementsData.microwaveCount >= 15 && GetAchievementsData.microwa
            {
                a18.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveSilver
                a18Txt.Text = "Microwave Silver";
            }
            else if (GetAchievementsData.microwaveCount >= 25)
            {
                a18.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveGold.p
                a18Txt.Text = "Microwave Gold";
            }

            if (GetAchievementsData.reBatteriesCount >= 5 && GetAchievementsData.reBatteries
            {
                a19.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesBron
                a19Txt.Text = "Reusable Batteries Bronze";
            }
            else if (GetAchievementsData.reBatteriesCount >= 15 && GetAchievementsData.reBat
25)
            {
                a19.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesSilv
                a19Txt.Text = "Reusable Batteries Silver";
            }
            else if (GetAchievementsData.reBatteriesCount >= 25)
            {
                a19.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesGold
                a19Txt.Text = "Reusable Batteries Gold";
            }

            if (GetAchievementsData.offSocketCount >= 5 && GetAchievementsData.offSocketCoun
            {
                a20.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.socketBronze.pn
                a20Txt.Text = "Socket Off Bronze";
            }
            else if (GetAchievementsData.offSocketCount >= 15 && GetAchievementsData.offSock
            {
                a20.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.socketSilver.pn
                a20Txt.Text = "Socket Off Silver";
            }
            else if (GetAchievementsData.offSocketCount >= 25)
            {
                a20.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.socketGold.png"
                a20Txt.Text = "Socket Off Gold";
            }

            if (GetAchievementsData.fullWasherCount >= 5 && GetAchievementsData.fullWasherCo
            {
                a21.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineB
                a21Txt.Text = "Washing Machine Bronze";
            }
            else if (GetAchievementsData.fullWasherCount >= 15 && GetAchievementsData.fullWa
            {
                a21.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineS
                a21Txt.Text = "Washing Machine Silver";
            }
            else if (GetAchievementsData.fullWasherCount >= 25)
            {
                a21.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineG
                a21Txt.Text = "Washing Machine Gold";
            }

            if (GetAchievementsData.carpoolCount >= 5 && GetAchievementsData.carpoolCount <
            {
                a22.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolBronze.p
                a22Txt.Text = "Carpool Bronze";
            }
            else if (GetAchievementsData.carpoolCount >= 15 && GetAchievementsData.carpoolCo
            {
                a22.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolSilver.p
                a22Txt.Text = "Carpool Silver";
            }
            else if (GetAchievementsData.carpoolCount >= 25)
            {
                a22.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolGold.png
                a22Txt.Text = "Carpool Gold";
            }

            if (GetAchievementsData.cycleCount >= 5 && GetAchievementsData.cycleCount < 15)
            {
                a23.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleBronze.png
                a23Txt.Text = "Cycle Bronze";
            }
            else if (GetAchievementsData.cycleCount >= 15 && GetAchievementsData.cycleCount
            {
                a23.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleSilver.png
                a23Txt.Text = "Cycle Silver";
            }
            else if (GetAchievementsData.cycleCount >= 25)
            {
                a23.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleGold.png")
                a23Txt.Text = "Cycle Gold";
            }

            if (GetAchievementsData.ecoCarCount >= 5 && GetAchievementsData.ecoCarCount < 15
            {
                a24.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarBronze.png"
                a24Txt.Text = "Eco Car Bronze";
            }
            else if (GetAchievementsData.ecoCarCount >= 15 && GetAchievementsData.ecoCarCoun
            {
                a24.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarSilver.png"
                a24Txt.Text = "Eco Car Silver";
            }
            else if (GetAchievementsData.ecoCarCount >= 25)
            {
                a24.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarGold.png");
                a24Txt.Text = "Eco Car Gold";
            }

            if (GetAchievementsData.transportCount >= 5 && GetAchievementsData.transportCoun
            {
                a25.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbBronze.png");
                a25Txt.Text = "Public Transport Bronze";
            }
            else if (GetAchievementsData.transportCount >= 15 && GetAchievementsData.transpo
            {
                a25.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbSilver.png");
                a25Txt.Text = "Public Transport Silver";
            }
            else if (GetAchievementsData.transportCount >= 25)
            {
                a25.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbGold.png");
                a25Txt.Text = "Public Transport Gold";
            }

            if (GetAchievementsData.walkCount >= 5 && GetAchievementsData.walkCount < 15)
            {
                a26.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkBronze.png"
                a26Txt.Text = "Walk Bronze";
            }
            else if (GetAchievementsData.walkCount >= 15 && GetAchievementsData.walkCount <
            {
                a26.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkSilver.png"
                a26Txt.Text = "Walk Silver";
            }
            else if (GetAchievementsData.walkCount >= 25)
            {
                a26.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkGold.png");
                a26Txt.Text = "Walk Gold";
            }

            if (GetAchievementsData.applianceCount >= 5 && GetAchievementsData.applianceCoun
            {
                a27.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceBron
                a27Txt.Text = "Eco Appliance Bronze";
            }
            else if (GetAchievementsData.applianceCount >= 15 && GetAchievementsData.applian
            {
                a27.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceSilv
                a27Txt.Text = "Eco Appliance Silver";
            }
            else if (GetAchievementsData.applianceCount >= 25)
            {
                a27.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceGold
                a27Txt.Text = "Eco Appliance Gold";
            }

            if (GetAchievementsData.foodCount >= 5 && GetAchievementsData.foodCount < 15)
            {
                a28.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkBronze.pn
                a28Txt.Text = "Food In Bulk Bronze";
            }
            else if (GetAchievementsData.foodCount >= 15 && GetAchievementsData.foodCount <
            {
                a28.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkSilver.pn
                a28Txt.Text = "Food In Bulk Silver";
            }
            else if (GetAchievementsData.foodCount >= 25)
            {
                a28.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkGold.png"
                a28Txt.Text = "Food In Bulk Gold";
            }

            if (GetAchievementsData.clothesCount >= 5 && GetAchievementsData.clothesCount < 
            {
                a29.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalBronze
                a29Txt.Text = "Ethical Clothing Bronze";
            }
            else if (GetAchievementsData.clothesCount >= 15 && GetAchievementsData.clothesCo
            {
                a29.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalSilver
                a29Txt.Text = "Ethical Clothing Silver";
            }
            else if (GetAchievementsData.clothesCount >= 25)
            {
                a29.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalGold.p
                a29Txt.Text = "Ethical Clothing Gold";
            }

            if (GetAchievementsData.localCount >= 5 && GetAchievementsData.localCount < 15)
            {
                a30.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localBronze.p
                a30Txt.Text = "Buy Local Bronze";
            }
            else if (GetAchievementsData.localCount >= 15 && GetAchievementsData.localCount 
            {
                a30.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localSilver.p
                a30Txt.Text = "Buy Local Silver";
            }
            else if (GetAchievementsData.localCount >= 25)
            {
                a30.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localGold.png
                a30Txt.Text = "Buy Local Gold";
            }

            if (GetAchievementsData.clothNapkinCount >= 5 && GetAchievementsData.clothNapkin
            {
                a31.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinBronze.
                a31Txt.Text = "Napkin Bronze";
            }
            else if (GetAchievementsData.clothNapkinCount >= 15 && GetAchievementsData.cloth
25)
            {
                a31.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinSilver.
                a31Txt.Text = "Napkin Silver";
            }
            else if (GetAchievementsData.clothNapkinCount >= 25)
            {
                a31.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinGold.pn
                a31Txt.Text = "Napkin Gold";
            }

            if (GetAchievementsData.productCount >= 5 && GetAchievementsData.productCount <
            {
                a32.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productBronze
                a32Txt.Text = "Eco Product Bronze";
            }
            else if (GetAchievementsData.productCount >= 15 && GetAchievementsData.productCo
            {
                a32.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productSilver
                a32Txt.Text = "Eco Product Silver";
            }
            else if (GetAchievementsData.productCount >= 25)
            {
                a32.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productGold.p
                a32Txt.Text = "Eco Product Gold";
            }

            if (GetAchievementsData.reBagCount >= 5 && GetAchievementsData.reBagCount < 15)
            {
                a33.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagBronze.p
                a33Txt.Text = "Reusable Bag Bronze";
            }
            else if (GetAchievementsData.reBagCount >= 15 && GetAchievementsData.reBagCount
            {
                a33.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagSilver.p
                a33Txt.Text = "Reusable Bag Silver";
            }
            else if (GetAchievementsData.reBagCount >= 25)
            {
                a33.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagGold.png
                a33Txt.Text = "Reusable Bag Gold";
            }

            if (GetAchievementsData.looseLeafCount >= 5 && GetAchievementsData.looseLeafCoun
            {
                a34.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaBronze.png
                a34Txt.Text = "Loose Leaf Bronze";
            }
            else if (GetAchievementsData.looseLeafCount >= 15 && GetAchievementsData.looseLe
            {
                a34.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaSilver.png
                a34Txt.Text = "Loose Leaf Silver";
            }
            else if (GetAchievementsData.looseLeafCount >= 25)
            {
                a34.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaGold.png")
                a34Txt.Text = "Loose Leaf Gold";
            }

            if (GetAchievementsData.toothbrushCount >= 5 && GetAchievementsData.toothbrushCo
            {
                a35.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushBro
                a35Txt.Text = "Eco Brush Bronze";
            }
            else if (GetAchievementsData.toothbrushCount >= 15 && GetAchievementsData.toothb
            {
                a35.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushSil
                a35Txt.Text = "Eco Brush Silver";
            }
            else if (GetAchievementsData.toothbrushCount >= 25)
            {
                a35.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushGol
                a35Txt.Text = "Eco Brush Gold";
            }

            if (GetAchievementsData.clothTowelCount >= 5 && GetAchievementsData.clothTowelCo
            {
                a36.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelBronze.p
                a36Txt.Text = "Towel Bronze";
            }
            else if (GetAchievementsData.clothTowelCount >= 15 && GetAchievementsData.clothT
            {
                a36.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelSilver.p
                a36Txt.Text = "Towel Silver";
            }
            else if (GetAchievementsData.clothTowelCount >= 25)
            {
                a36.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelGold.png
                a36Txt.Text = "Towel Gold";
            }

            if (GetAchievementsData.reWaterCount >= 5 && GetAchievementsData.reWaterCount <
            {
                a37.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleBronze.png
                a37Txt.Text = "Reusable Bottle Bronze";
            }
            else if (GetAchievementsData.reWaterCount >= 15 && GetAchievementsData.reWaterCo
            {
                a37.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleSilver.png
                a37Txt.Text = "Reusable Bottle Silver";
            }
            else if (GetAchievementsData.reWaterCount >= 25)
            {
                a37.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleGold.png")
                a37Txt.Text = "Reusable Bottle Gold";
            }

            if (GetAchievementsData.showerBucketCount >= 5 && GetAchievementsData.showerBuck
            {
                a38.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketBronze.png
                a38Txt.Text = "Shower Bucket Bronze";
            }
            else if (GetAchievementsData.showerBucketCount >= 15 && GetAchievementsData.show
25)
            {
                a38.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketSilver.png
                a38Txt.Text = "Shower Bucket Silver";
            }
            else if (GetAchievementsData.showerBucketCount >= 25)
            {
                a38.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketGold.png")
                a38Txt.Text = "Shower Bucket Gold";
            }

            if (GetAchievementsData.airOutCount >= 5 && GetAchievementsData.airOutCount < 15
            {
                a39.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airBronze.png");
                a39Txt.Text = "Ait Out Bronze";
            }
            else if (GetAchievementsData.airOutCount >= 15 && GetAchievementsData.airOutCoun
            {
                a39.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airSilver.png");
                a39Txt.Text = "Ait Out Silver";
            }
            else if (GetAchievementsData.airOutCount >= 25)
            {
                a39.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airGold.png");
                a39Txt.Text = "Ait Out Gold";
            }

            if (GetAchievementsData.toiletFlushCount >= 5 && GetAchievementsData.toiletFlush
            {
                a40.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushBronze.png")
                a40Txt.Text = "Save Flush Bronze";
            }
            else if (GetAchievementsData.toiletFlushCount >= 15 && GetAchievementsData.toile
25)
            {
                a40.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushSilver.png")
                a40Txt.Text = "Save Flush Silver";
            }
            else if (GetAchievementsData.toiletFlushCount >= 25)
            {
                a40.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushGold.png");
                a40Txt.Text = "Save Flush Gold";
            }

            if (GetAchievementsData.nonHarmCount >= 5 && GetAchievementsData.nonHarmCount <
            {
                a41.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.nonHarmfulBronze.
                a41Txt.Text = "Non Harmful Bronze";
            }
            else if (GetAchievementsData.nonHarmCount >= 15 && GetAchievementsData.nonHarmCo
            {
                a41.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.nonHarmfulSilver.
                a41Txt.Text = "Non Harmful Silver";
            }
            else if (GetAchievementsData.nonHarmCount >= 25)
            {
                a41.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.nonHarmfulGold.pn
                a41Txt.Text = "Non Harmful Gold";
            }

            if (GetAchievementsData.outsideCount >= 5 && GetAchievementsData.outsideCount <
            {
                a42.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.outsideBronze.png
                a42Txt.Text = "Go Outside Bronze";
            }
            else if (GetAchievementsData.outsideCount >= 15 && GetAchievementsData.outsideCo
            {
                a42.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.outsideSilver.png
                a42Txt.Text = "Go Outside Silver";
            }
            else if (GetAchievementsData.outsideCount >= 25)
            {
                a42.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.outsideGold.png")
                a42Txt.Text = "Go Outside Gold";
            }

            if (GetAchievementsData.plantIntoHomeCount >= 5 && GetAchievementsData.plantInto
            {
                a43.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.plantBronze.png")
                a43Txt.Text = "Home Plant Bronze";
            }
            else if (GetAchievementsData.plantIntoHomeCount >= 15 && GetAchievementsData.pla
< 25)
            {
                a43.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.plantSilver.png")
                a43Txt.Text = "Home Plant Silver";
            }
            else if (GetAchievementsData.plantIntoHomeCount >= 25)
            {
                a43.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.plantGold.png");
                a43Txt.Text = "Home Plant Gold";
            }

            if (GetAchievementsData.plantBushCount >= 5 && GetAchievementsData.plantBushCoun
            {
                a44.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.bushBronze.pn
                a41Txt.Text = "Bush Planting Bronze";
            }
            else if (GetAchievementsData.plantBushCount >= 15 && GetAchievementsData.plantBu
            {
                a44.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.bushSilver.pn
                a41Txt.Text = "Bush Planting Silver";
            }
            else if (GetAchievementsData.plantBushCount >= 25)
            {
                a44.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.bushGold.png"
                a41Txt.Text = "Bush Planting Gold";
            }

            if (GetAchievementsData.campingCount >= 5 && GetAchievementsData.campingCount <
            {
                a45.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingBronze
                a45Txt.Text = "Camping Bronze";
            }
            else if (GetAchievementsData.campingCount >= 15 && GetAchievementsData.campingCo
            {
                a45.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingSilver
                a45Txt.Text = "Camping Silver";
            }
            else if (GetAchievementsData.campingCount >= 25)
            {
                a45.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingGold.p
                a45Txt.Text = "Camping Gold";
            }

            if (GetAchievementsData.plantFlowerCount >= 5 && GetAchievementsData.plantFlower
            {
                a46.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerBronze.
                a46Txt.Text = "Flower Planting Bronze";
            }
            else if (GetAchievementsData.plantFlowerCount >= 15 && GetAchievementsData.plant
25)
            {
                a46.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerSilver.
                a46Txt.Text = "Flower Planting Silver";
            }
            else if (GetAchievementsData.plantFlowerCount >= 25)
            {
                a46.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerGold.pn
                a46Txt.Text = "Flower Planting Gold";
            }

            if (GetAchievementsData.picnicCount >= 5 && GetAchievementsData.picnicCount < 15
            {
                a47.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicBronze.
                a47Txt.Text = "Picnic Bronze";
            }
            else if (GetAchievementsData.picnicCount >= 15 && GetAchievementsData.picnicCoun
            {
                a47.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicSilver.
                a47Txt.Text = "Picnic Silver";
            }
            else if (GetAchievementsData.picnicCount >= 25)
            {
                a47.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicGold.pn
                a47Txt.Text = "Picnic Gold";
            }

            if (GetAchievementsData.scoopCount >= 5 && GetAchievementsData.scoopCount < 15)
            {
                a48.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.scoopBronze.p
                a48Txt.Text = "Scoop Poop Bronze";
            }
            else if (GetAchievementsData.scoopCount >= 15 && GetAchievementsData.scoopCount
            {
                a48.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.scoopSilver.p
                a48Txt.Text = "Scoop Poop Silver";
            }
            else if (GetAchievementsData.scoopCount >= 25)
            {
                a48.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.scoopGold.png
                a48Txt.Text = "Scoop Poop Gold";
            }

            if (GetAchievementsData.plantTreeCount >= 5 && GetAchievementsData.plantTreeCoun
            {
                a49.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.treeBronze.pn
                a49Txt.Text = "Tree Planting Bronze";
            }
            else if (GetAchievementsData.plantTreeCount >= 15 && GetAchievementsData.plantTr
            {
                a49.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.treeSilver.pn
                a49Txt.Text = "Tree Planting Silver";
            }
            else if (GetAchievementsData.plantTreeCount >= 25)
            {
                a49.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.treeGold.png"
                a49Txt.Text = "Tree Planting Gold";
            }

            if (GetAchievementsData.communityCount >= 5 && GetAchievementsData.communityCoun
            {
                a50.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.communityBro
                a50Txt.Text = "Community Bronze";
            }
            else if (GetAchievementsData.communityCount >= 15 && GetAchievementsData.communi
            {
                a50.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.communitySil
                a50Txt.Text = "Community Silver";
            }
            else if (GetAchievementsData.communityCount >= 25)
            {
                a50.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.communityGol
                a50Txt.Text = "Community Gold";
            }

            if (GetAchievementsData.donateCount >= 5 && GetAchievementsData.donateCount < 15
            {
                a51.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.donateBronze
                a51Txt.Text = "Donations Bronze";
            }
            else if (GetAchievementsData.donateCount >= 15 && GetAchievementsData.donateCoun
            {
                a51.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.donateSilver
                a51Txt.Text = "Donations Silver";
            }
            else if (GetAchievementsData.donateCount >= 25)
            {
                a51.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.donateGold.p
                a51Txt.Text = "Donations Gold";
            }

            if (GetAchievementsData.bioBinBagsCount >= 5 && GetAchievementsData.bioBinBagsCo
            {
                a52.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsBronze.png")
                a52Txt.Text = "Bio Bin Bag Bronze";
            }
            else if (GetAchievementsData.bioBinBagsCount >= 15 && GetAchievementsData.bioBin
            {
                a52.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsSilver.png")
                a52Txt.Text = "Bio Bin Bag Silver";
            }
            else if (GetAchievementsData.bioBinBagsCount >= 25)
            {
                a52.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsGold.png");
                a52Txt.Text = "Bio Bin Bag Gold";
            }

            if (GetAchievementsData.billsCount >= 5 && GetAchievementsData.billsCount < 15)
            {
                a53.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsBronze.png"
                a53Txt.Text = "Online Bills Bronze";
            }
            else if (GetAchievementsData.billsCount >= 15 && GetAchievementsData.billsCount
            {
                a53.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsSilver.png"
                a53Txt.Text = "Online Bills Silver";
            }
            else if (GetAchievementsData.billsCount >= 25)
            {
                a53.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsGold.png");
                a53Txt.Text = "Online Bills Gold";
            }

            if (GetAchievementsData.recyclingBinCount >= 5 && GetAchievementsData.recyclingB
            {
                a54.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binBronze.png");
                a54Txt.Text = "Recycling Bronze";
            }
            else if (GetAchievementsData.recyclingBinCount >= 15 && GetAchievementsData.recy
25)
            {
                a54.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binSilver.png");
                a54Txt.Text = "Recycling Silver";
            }
            else if (GetAchievementsData.recyclingBinCount >= 25)
            {
                a54.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binGold.png");
                a54Txt.Text = "Recycling Gold";
            }

            if (GetAchievementsData.compostCount >= 5 && GetAchievementsData.compostCount <
            {
                a55.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.compostBronze.pn
                a55Txt.Text = "Composting Bronze";
            }
            else if (GetAchievementsData.compostCount >= 15 && GetAchievementsData.compostCo
            {
                a55.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.compostSilver.pn
                a55Txt.Text = "Composting Silver";
            }
            else if (GetAchievementsData.compostCount >= 25)
            {
                a55.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.compostGold.png"
                a55Txt.Text = "Composting Gold";
            }

            if (GetAchievementsData.offElectronicsCount >= 5 && GetAchievementsData.offElect
15)
            {
                a56.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.offBronze.png");
                a56Txt.Text = "Electronics Off Bronze";
            }
            else if (GetAchievementsData.offElectronicsCount >= 15 &&
GetAchievementsData.offElectronicsCount < 25)
            {
                a56.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.offSilver.png");
                a56Txt.Text = "Electronics Off Silver";
            }
            else if (GetAchievementsData.offElectronicsCount >= 25)
            {
                a56.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.offGold.png");
                a56Txt.Text = "Electronics Off Gold";
            }

            if (GetAchievementsData.paperCount >= 5 && GetAchievementsData.paperCount < 15)
            {
                a57.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.paperBronze.png")
                a57Txt.Text = " Paper Bronze";
            }
            else if (GetAchievementsData.paperCount >= 15 && GetAchievementsData.paperCount
            {
                a57.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.paperSilver.png")
                a57Txt.Text = " Paper Silver";
            }
            else if (GetAchievementsData.paperCount >= 25)
            {
                a57.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.paperGold.png");
                a57Txt.Text = " Paper Gold";
            }

            if (GetAchievementsData.fixCount >= 5 && GetAchievementsData.fixCount < 15)
            {
                a58.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixBronze.png
                a58Txt.Text = "Fixed Bronze";
            }
            else if (GetAchievementsData.fixCount >= 15 && GetAchievementsData.fixCount < 25
            {
                a58.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixSilver.png
                a58Txt.Text = "Fixed Silver";
            }
            else if (GetAchievementsData.fixCount >= 25)
            {
                a58.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixGold.png")
                a58Txt.Text = "Fixed Gold";
            }

            if (GetAchievementsData.efficientThermostatCount >= 1)
            {
                a59.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.thermos
                a59Txt.Text = "Efficient Thermostat Set";
            }

            if (GetAchievementsData.insulateWaterCount >= 1)
            {
                a60.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.waterTa
                a60Txt.Text = "Insulated Water Tank";
            }

            if (GetAchievementsData.isolateHomeCount >= 1)
            {
                a61.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.isolate
                a61Txt.Text = "Insulated Home";
            }

            if (GetAchievementsData.ledLightBulbCount >= 1)
            {
                a62.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.ledBulb
                a62Txt.Text = "Led Lights Installed";
            }

            if (GetAchievementsData.fridgeCount >= 1)
            {
                a63.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.turnDow
                a63Txt.Text = "Turn Down The Fridge";
            }

            if (GetAchievementsData.draftSealCount >= 1)
            {
                a64.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.sealDra
                a64Txt.Text = "Drafts Sealed";
            }

            if (GetAchievementsData.ductSealCount >= 1)
            {
                a65.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.sealDuc
                a65Txt.Text = "Ducts Sealed";
            }

            if (GetAchievementsData.solarPanelCount >= 1)
            {
                a66.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.solar.p
                a66Txt.Text = "Installed Solar Panel";
            }

            if (GetAchievementsData.reusableCount >= 1)
            {
                a67.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Shopping.reWat
                a67Txt.Text = "Purchased Reusable Bottle";
            }

            if (GetAchievementsData.reBatCount >= 1)
            {
                a68.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Shopping.reBat
                a68Txt.Text = "Purchased Reusable Batteries";
            }

            if (GetAchievementsData.cisternCount >= 1)
            {
                a69.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.Cistern.
                a69Txt.Text = "Cistern Displacement System Installed";
            }

            if (GetAchievementsData.rainBarrelCount >= 1)
            {
                a70.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.rainBarr
                a70Txt.Text = "Rain Barrel Set Up";
            }

            if (GetAchievementsData.wSShowerHeadCount >= 1)
            {
                a71.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.showerHe
                a71Txt.Text = "Water Saving Shower Head Installed";
            }

            if (GetAchievementsData.fruitGardenCount >= 1)
            {
                a72.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Outdoors.fruit
                a72Txt.Text = "Fruit Garden Set Up";
            }

            if (GetAchievementsData.herbGardenCount >= 1)
            {
                a73.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Outdoors.herb.
                a73Txt.Text = "Herb Garden Set Up";
            }

            if (GetAchievementsData.vegetableGardenCount >= 1)
            {
                a74.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Outdoors.veg.p
                a74Txt.Text = "Vegetable Garden Set Up";
            }

            if (GetAchievementsData.birdFeederCount >= 1)
            {
                a75.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Outdoors.birdF
                a75Txt.Text = "Bird Feeder Set Up";
            }

            if (GetAchievementsData.createGroupCount >= 1)
            {
                a76.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Community.crea
                a76Txt.Text = "Community Set Up";
            }

            if (GetAchievementsData.groupCount >= 1)
            {
                a77.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Community.join
                a77Txt.Text = "Joined A Community";
            }

            if (GetAchievementsData.shareCount >= 1)
            {
                a78.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Community.shar
                a78Txt.Text = "App Shared";
            }

            if (GetAchievementsData.awarenessCount >= 1)
            {
                a79.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Community.spre
                a79Txt.Text = "Awareness Spread";
            }

            if (GetAchievementsData.setUpRecyclingBinCount >= 1)
            {
                a80.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Waste.bins.png
                a80Txt.Text = "Recycling Bins Set Up";
            }

            if (GetAchievementsData.remoteWorkCount >= 1)
            {
                a81.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Work.remote.pn
                a81Txt.Text = "Working Remotely";
            }
        }
    }
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pages:PopupPage x:Class="Application_Green_Quake.Views.ProfilePage.BadgePopUp"
                 xmlns="http://xamarin.com/schemas/2014/forms"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 xmlns:pages="clr-
namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
                 xmlns:local="clr-namespace:Application_Green_Quake.Models"
                 BackgroundColor="#002a1e">

    <Grid VerticalOptions="FillAndExpand" RowSpacing="0" BackgroundColor="#002a1e">
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <StackLayout BackgroundColor="#002a1e"
                     VerticalOptions="End">
            <Label x:Name="badgeHeading" Text="Unranked" FontSize="25" TextColor="White"
HorizontalTextAlignment="Center" />
            <Label x:Name="badgeSubHeading" Text="" FontSize="15" TextColor="White"
HorizontalTextAlignment="Center" />
        </StackLayout>

         <Image Grid.Row="1"
                Aspect="AspectFill"
                BackgroundColor="#002a1e"
                Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}"
                x:Name="badge"/>

         <StackLayout Grid.Row="2"
                      BackgroundColor="#002a1e">
            <Label x:Name="badgeTxt" Text="Next Rank: After 1 Action Log" FontSize="25"
TextColor="White" HorizontalTextAlignment="Center"/>
         </StackLayout>

    </Grid>
</pages:PopupPage>
```

```
/*! \class The BadgePopUp View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the BadgePopUp View Class. This class is the popup that appears when
badge is tapped.
 *
 */
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.ProfilePage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class BadgePopUp
    {
        int num = 0;
        int badgeState= 0;
        /** The BadgePopUp Constructor
        @param number is used to specify which badge it is
        @param stage is used to specify what stage the badge is in
        */
        public BadgePopUp(int number, int stage)
        {
            num = number;
            badgeState = stage;
            InitializeComponent();
            OnAppearing();
        }
        /** This function is called before the page is displayed. It displays the correct
information and badge based on what was passed into the
         * constructor
        */
        protected override void OnAppearing()
        {

            if (num == 1 && badgeState == 1)
            {
                badgeHeading.Text = "Habit Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 1 && badgeState == 2)
            {
                badgeHeading.Text = "Habit Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 1 && badgeState == 3)
            {
                badgeHeading.Text = "Habit Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 1 && badgeState == 4)
            {
                badgeHeading.Text = "Habit Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 1 && badgeState == 5)
            {
                badgeHeading.Text = "Habit Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 1 && badgeState == 6)
            {
                badgeHeading.Text = "Habit Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 2 && badgeState == 1)
            {
                badgeHeading.Text = "Advanced Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 2 && badgeState == 2)
            {
                badgeHeading.Text = "Advanced Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 2 && badgeState == 3)
            {
                badgeHeading.Text = "Advanced Habit Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 2 && badgeState == 4)
            {
                badgeHeading.Text = "Advanced Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 2 && badgeState == 5)
            {
                badgeHeading.Text = "Advanced Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 2 && badgeState == 6)
            {
                badgeHeading.Text = "Advanced Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 3 && badgeState == 1)
            {
                badgeHeading.Text = "Community Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 3 && badgeState == 2)
            {
                badgeHeading.Text = "Community Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 3 && badgeState == 3)
            {
                badgeHeading.Text = "Community Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityThree.png"
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 3 && badgeState == 4)
            {
                badgeHeading.Text = "Community Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFour.png")
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 3 && badgeState == 5)
            {
                badgeHeading.Text = "Community Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFive.png")
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 3 && badgeState == 6)
            {
                badgeHeading.Text = "Community Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communitySix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 4 && badgeState == 1)
            {
                badgeHeading.Text = "Energy Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 4 && badgeState == 2)
            {
                badgeHeading.Text = "Energy Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 4 && badgeState == 3)
            {
                badgeHeading.Text = "Energy Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 4 && badgeState == 4)
            {
                badgeHeading.Text = "Energy Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 4 && badgeState == 5)
            {
                badgeHeading.Text = "Energy Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 4 && badgeState == 6)
            {
                badgeHeading.Text = "Energy Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energySix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 5 && badgeState == 1)
            {
                badgeHeading.Text = "Food And Drink Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 5 && badgeState == 2)
            {
                badgeHeading.Text = "Food And Drink Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 5 && badgeState == 3)
            {
                badgeHeading.Text = "Food And Drink Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 5 && badgeState == 4)
            {
                badgeHeading.Text = "Food And Drink Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 5 && badgeState == 5)
            {
                badgeHeading.Text = "Food And Drink Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 5 && badgeState == 6)
            {
                badgeHeading.Text = "Food And Drink Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 6 && badgeState == 1)
            {
                badgeHeading.Text = "Home Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 6 && badgeState == 2)
            {
                badgeHeading.Text = "Home Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 6 && badgeState == 3)
            {
                badgeHeading.Text = "Home Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 6 && badgeState == 4)
            {
                badgeHeading.Text = "Home Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 6 && badgeState == 5)
            {
                badgeHeading.Text = "Home Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 6 && badgeState == 6)
            {
                badgeHeading.Text = "Home Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 7 && badgeState == 1)
            {
                badgeHeading.Text = "Outdoors Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 7 && badgeState == 2)
            {
                badgeHeading.Text = "Outdoors Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 7 && badgeState == 3)
            {
                badgeHeading.Text = "Outdoors Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 7 && badgeState == 4)
            {
                badgeHeading.Text = "Outdoors Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 7 && badgeState == 5)
            {
                badgeHeading.Text = "Outdoors Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 7 && badgeState == 6)
            {
                badgeHeading.Text = "Outdoors Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 8 && badgeState == 1)
            {
                badgeHeading.Text = "Shopping Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 8 && badgeState == 2)
            {
                badgeHeading.Text = "Shopping Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 8 && badgeState == 3)
            {
                badgeHeading.Text = "Shopping Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 8 && badgeState == 4)
            {
                badgeHeading.Text = "Shopping Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 8 && badgeState == 5)
            {
                badgeHeading.Text = "Shopping Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 8 && badgeState == 6)
            {
                badgeHeading.Text = "Shopping Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 9 && badgeState == 1)
            {
                badgeHeading.Text = "Travel Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 9 && badgeState == 2)
            {
                badgeHeading.Text = "Travel Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 9 && badgeState == 3)
            {
                badgeHeading.Text = "Travel Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 9 && badgeState == 4)
            {
                badgeHeading.Text = "Travel Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 9 && badgeState == 5)
            {
                badgeHeading.Text = "Travel Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 9 && badgeState == 6)
            {
                badgeHeading.Text = "Travel Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 10 && badgeState == 1)
            {
                badgeHeading.Text = "Waste Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 10 && badgeState == 2)
            {
                badgeHeading.Text = "Waste Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 10 && badgeState == 3)
            {
                badgeHeading.Text = "Waste Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 10 && badgeState == 4)
            {
                badgeHeading.Text = "Waste Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 10 && badgeState == 5)
            {
                badgeHeading.Text = "Waste Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 10 && badgeState == 6)
            {
                badgeHeading.Text = "Waste Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 11 && badgeState == 1)
            {
                badgeHeading.Text = "Water Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 11 && badgeState == 2)
            {
                badgeHeading.Text = "Water Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 11 && badgeState == 3)
            {
                badgeHeading.Text = "Water Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 11 && badgeState == 4)
            {
                badgeHeading.Text = "Water Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 11 && badgeState == 5)
            {
                badgeHeading.Text = "Water Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 11 && badgeState == 6)
            {
                badgeHeading.Text = "Water Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }

            if (num == 12 && badgeState == 1)
            {
                badgeHeading.Text = "Work Novice";
                badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workOne.png");
                badgeTxt.Text = "Next Rank: After 5 Action Logs";
            }
            else if (num == 12 && badgeState == 2)
            {
                badgeHeading.Text = "Work Apprentice";
                badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workTwo.png");
                badgeTxt.Text = "Next Rank: After 10 Action Logs";
            }
            else if (num == 12 && badgeState == 3)
            {
                badgeHeading.Text = "Work Adept";
                badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workThree.png");
                badgeTxt.Text = "Next Rank: After 25 Action Logs";
            }
            else if (num == 12 && badgeState == 4)
            {
                badgeHeading.Text = "Work Expert";
                badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFour.png");
                badgeTxt.Text = "Next Rank: After 50 Action Logs";
            }
            else if (num == 12 && badgeState == 5)
            {
                badgeHeading.Text = "Work Master";
                badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFive.png");
                badgeTxt.Text = "Next Rank: After 100 Action Logs";
            }
            else if (num == 12 && badgeState == 6)
            {
                badgeHeading.Text = "Work Legend";
                badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
                badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workSix.png");
                badgeTxt.Text = "Max Rank Reached!";
            }
```

```
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.ProfilePage.Badges"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">
    <ContentPage.Content>
        <ScrollView>
            <Grid BackgroundColor="#002a1e" HorizontalOptions="Fill" VerticalOptions="Fill" Padding="10,10,10,10">
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>

                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <StackLayout Grid.Column="0" Grid.Row="0">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpOne"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a1" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a1Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center"/>
                </StackLayout>

                <StackLayout Grid.Column="1" Grid.Row="0">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpTwo"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a2" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a2Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center"/>
                </StackLayout>

                <StackLayout Grid.Column="2" Grid.Row="0">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpThree"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a3" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>
                    <Label x:Name="a3Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center" />
                </StackLayout>

                <StackLayout Grid.Column="3" Grid.Row="0">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpFour"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a4" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>

                    </Frame>

                    <Label x:Name="a4Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center" />
                </StackLayout>

                <StackLayout Grid.Column="0" Grid.Row="1">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpFive"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a5" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>

                    </Frame>

                    <Label x:Name="a5Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center"/>
                </StackLayout>

                <StackLayout Grid.Column="1" Grid.Row="1">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpSix"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a6" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a6Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center"/>
                </StackLayout>

                <StackLayout Grid.Column="2" Grid.Row="1">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpSeven"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a7" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a7Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center" />
                </StackLayout>

                <StackLayout Grid.Column="3" Grid.Row="1">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpEight"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a8" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a8Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center" />
                </StackLayout>

                <StackLayout Grid.Column="0" Grid.Row="2">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpNine"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a9" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a9Txt" Text="Locked " TextColor="White" HorizontalTextAlignment="Center"/>
                </StackLayout>

                <StackLayout Grid.Column="1" Grid.Row="2">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpTen"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a10" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a10Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center"/>
                </StackLayout>

                <StackLayout Grid.Column="2" Grid.Row="2">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpEleven"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a11" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
                    </Frame>

                    <Label x:Name="a11Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center" />
                </StackLayout>

                <StackLayout Grid.Column="3" Grid.Row="2">
                    <StackLayout.GestureRecognizers>
                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpTwelve"/>
                    </StackLayout.GestureRecognizers>
                    <Frame CornerRadius="60" HorizontalOptions="Start" VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
                        <Image x:Name="a12" Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand" />
                    </Frame>

                    <Label x:Name="a12Txt" Text="Locked" TextColor="White" HorizontalTextAlignment="Center" />
                </StackLayout>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The Badges View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, 
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the Badges View Class. This class is the class that displays all the 
Badges on the Badges page.
 *
 */
using Application_Green_Quake.ViewModels;
using Rg.Plugins.Popup.Services;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.ProfilePage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Badges : ContentPage
    {
        int habitsStage = 0;
        int advancedStage = 0;
        int communityStage = 0;
        int energyStage = 0;
        int foodDrinkStage = 0;
        int homeStage = 0;
        int outdoorsStage = 0;
        int shoppingStage = 0;
        int travelStage = 0;
        int wasteStage = 0;
        int waterStage = 0;
        int workStage = 0;
        public Badges()
        {
            InitializeComponent();
            OnAppearing();
        }
        /** This function is called before the page is displayed. It displays the images as t
criteria are met
        */
        protected override void OnAppearing()
        {
            if (GetBadgeData.habitsLog > 0 && GetBadgeData.habitsLog < 5)
            {
                a1.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsOne.png");
                a1Txt.Text = "Habits Novice";
                habitsStage = 1;
            }
            else if (GetBadgeData.habitsLog >= 5 && GetBadgeData.habitsLog < 10)
            {
                a1.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsTwo.png");
                a1Txt.Text = "Habits Apprentice";
                habitsStage = 2;
            }
            else if (GetBadgeData.habitsLog >= 10 && GetBadgeData.habitsLog < 25)
            {
                a1.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsThree.png");
                a1Txt.Text = "Habits Adept";
                habitsStage = 3;
            }
            else if (GetBadgeData.habitsLog >= 25 && GetBadgeData.habitsLog < 50)
            {
                a1.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFour.png");
                a1Txt.Text = "Habits Expert";
                habitsStage = 4;
            }
            else if (GetBadgeData.habitsLog >= 50 && GetBadgeData.habitsLog < 100)
            {
                a1.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFive.png");
                a1Txt.Text = "Habits Master";
                habitsStage = 5;
            }
            else if (GetBadgeData.habitsLog >= 100)
            {
                a1.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsSix.png");
                a1Txt.Text = "Habits Legend";
                habitsStage = 6;
            }


            if (GetBadgeData.advancedLog > 0 && GetBadgeData.advancedLog < 5)
            {
                a2.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedOne.png");
                a2Txt.Text = "Advanced Novice";
                advancedStage = 1;
            }
            else if (GetBadgeData.advancedLog >= 5 && GetBadgeData.advancedLog < 10)
            {
                a2.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedTwo.png");
                a2Txt.Text = "Advanced Apprentice";
                advancedStage = 2;
            }
            else if (GetBadgeData.advancedLog >= 10 && GetBadgeData.advancedLog < 25)
            {
                a2.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedThree.png");
                a2Txt.Text = "Advanced Adept";
                advancedStage = 3;
            }
            else if (GetBadgeData.advancedLog >= 25 && GetBadgeData.advancedLog < 50)
            {
                a2.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFour.png");
                a2Txt.Text = "Advanced Expert";
                advancedStage = 4;
            }
            else if (GetBadgeData.advancedLog >= 50 && GetBadgeData.advancedLog < 100)
            {
                a2.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFive.png");
                a2Txt.Text = "Advanced Master";
                advancedStage = 5;
            }
            else if (GetBadgeData.advancedLog >= 100)
            {
                a2.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedSix.png");
                a2Txt.Text = "Advanced Legend";
                advancedStage = 6;
            }

            if (GetBadgeData.communityLog > 0 && GetBadgeData.communityLog < 5)
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityOne.png");
                a3Txt.Text = "Community Novice";
                communityStage = 1;
            }
            else if (GetBadgeData.communityLog >= 5 && GetBadgeData.communityLog < 10)
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityTwo.png");
                a3Txt.Text = "Community Apprentice";
                communityStage = 2;
            }
            else if (GetBadgeData.communityLog >= 10 && GetBadgeData.communityLog < 25)
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityThree.png"
                a3Txt.Text = "Community Adept";
                communityStage = 3;
            }
            else if (GetBadgeData.communityLog >= 25 && GetBadgeData.communityLog < 50)
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFour.png")
                a3Txt.Text = "Community Expert";
                communityStage = 4;
            }
            else if (GetBadgeData.communityLog >= 50 && GetBadgeData.communityLog < 100)
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFive.png")
                a3Txt.Text = "Community Master";
                communityStage = 5;
            }
            else if (GetBadgeData.communityLog >= 100)
            {
                a3.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communitySix.png");
                a3Txt.Text = "Community Legend";
                communityStage = 6;
            }

            if (GetBadgeData.energyLog > 0 && GetBadgeData.energyLog < 5)
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyOne.png");
                a4Txt.Text = "Energy Novice";
                energyStage = 1;
            }
            else if (GetBadgeData.energyLog >= 5 && GetBadgeData.energyLog < 10)
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyTwo.png");
                a4Txt.Text = "Energy Apprentice";
                energyStage = 2;
            }
            else if (GetBadgeData.energyLog >= 10 && GetBadgeData.energyLog < 25)
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyThree.png");
                a4Txt.Text = "Energy Adept";
                energyStage = 3;
            }
            else if (GetBadgeData.energyLog >= 25 && GetBadgeData.energyLog < 50)
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFour.png");
                a4Txt.Text = "Energy Expert";
                energyStage = 4;
            }
            else if (GetBadgeData.energyLog >= 50 && GetBadgeData.energyLog < 100)
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFive.png");
                a4Txt.Text = "Energy Master";
                energyStage = 5;
            }
            else if (GetBadgeData.energyLog >= 100)
            {
                a4.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energySix.png");
                a4Txt.Text = "Energy Legend";
                energyStage = 6;
            }

            if (GetBadgeData.foodDrinkLog > 0 && GetBadgeData.foodDrinkLog < 5)
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDOne.png");
                a5Txt.Text = "F & D Novice";
                foodDrinkStage = 1;
            }
            else if (GetBadgeData.foodDrinkLog >= 5 && GetBadgeData.foodDrinkLog < 10)
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDTwo.png");
                a5Txt.Text = "F & D Apprentice";
                foodDrinkStage = 2;
            }
            else if (GetBadgeData.foodDrinkLog >= 10 && GetBadgeData.foodDrinkLog < 25)
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDThree.png");
                a5Txt.Text = "F & D Adept";
                foodDrinkStage = 3;
            }
            else if (GetBadgeData.foodDrinkLog >= 25 && GetBadgeData.foodDrinkLog < 50)
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFour.png");
                a5Txt.Text = "F & D Expert";
                foodDrinkStage = 4;
            }
            else if (GetBadgeData.foodDrinkLog >= 50 && GetBadgeData.foodDrinkLog < 100)
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFive.png");
                a5Txt.Text = "F & D Master";
                foodDrinkStage = 5;
            }
            else if (GetBadgeData.foodDrinkLog >= 100)
            {
                a5.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDSix.png");
                a5Txt.Text = "F & D Legend";
                foodDrinkStage = 6;
            }

            if (GetBadgeData.homeLog > 0 && GetBadgeData.homeLog < 5)
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeOne.png");
                a6Txt.Text = "Home Novice";
                homeStage = 1;
            }
            else if (GetBadgeData.homeLog >= 5 && GetBadgeData.homeLog < 10)
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeTwo.png");
                a6Txt.Text = "Home Apprentice";
                homeStage = 2;
            }
            else if (GetBadgeData.homeLog >= 10 && GetBadgeData.homeLog < 25)
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeThree.png");
                a6Txt.Text = "Home Adept";
                homeStage = 3;
            }
            else if (GetBadgeData.homeLog >= 25 && GetBadgeData.homeLog < 50)
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFour.png");
                a6Txt.Text = "Home Expert";
                homeStage = 4;
            }
            else if (GetBadgeData.homeLog >= 50 && GetBadgeData.homeLog < 100)
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFive.png");
                a6Txt.Text = "Home Master";
                homeStage = 5;
            }
            else if (GetBadgeData.homeLog >= 100)
            {
                a6.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeSix.png");
                a6Txt.Text = "Home Legend";
                homeStage = 6;
            }

            if (GetBadgeData.outdoorsLog > 0 && GetBadgeData.outdoorsLog < 5)
            {
                a7.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsOne.png");
                a7Txt.Text = "Outdoors Novice";
                outdoorsStage = 1;
            }
            else if (GetBadgeData.outdoorsLog >= 5 && GetBadgeData.outdoorsLog < 10)
            {
                a7.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsTwo.png");
                a7Txt.Text = "Outdoors Apprentice";
                outdoorsStage = 2;
            }
            else if (GetBadgeData.outdoorsLog >= 10 && GetBadgeData.outdoorsLog < 25)
            {
                a7.Source = ImageSource.FromResource("Applica
tion_Green_Quake.Images.Badges.Outdoors.outdoorsThree.png");
                a7Txt.Text = "Outdoors Adept";
                outdoorsStage = 3;
            }
            else if (GetBadgeData.outdoorsLog >= 25 && GetBadgeData.outdoorsLog < 50)
            {
                a7.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFour.png");
                a7Txt.Text = "Outdoors Expert";
                outdoorsStage = 4;
            }
            else if (GetBadgeData.outdoorsLog >= 50 && GetBadgeData.outdoorsLog < 100)
            {
                a7.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFive.png");
                a7Txt.Text = "Outdoors Master";
                outdoorsStage = 5;
            }
            else if (GetBadgeData.outdoorsLog >= 100)
            {
                a7.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsSix.png");
                a7Txt.Text = "Outdoors Legend";
                outdoorsStage = 6;
            }

            if (GetBadgeData.shoppingLog > 0 && GetBadgeData.shoppingLog < 5)
            {
                a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingOne.png");
                a8Txt.Text = "Shopping Novice";
                shoppingStage = 1;
            }
            else if (GetBadgeData.shoppingLog >= 5 && GetBadgeData.shoppingLog < 10)
            {
                a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingTwo.png");
                a8Txt.Text = "Shopping Apprentice";
                shoppingStage = 2;
            }
            else if (GetBadgeData.shoppingLog >= 10 && GetBadgeData.shoppingLog < 25)
            {
                a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingThree.png");
                a8Txt.Text = "Shopping Adept";
                shoppingStage = 3;
            }
            else if (GetBadgeData.shoppingLog >= 25 && GetBadgeData.shoppingLog < 50)
            {
                a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFour.png");
                a8Txt.Text = "Shopping Expert";
                shoppingStage = 4;
            }
            else if (GetBadgeData.shoppingLog >= 50 && GetBadgeData.shoppingLog < 100)
            {
                a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFive.png");
```

```
                a8Txt.Text = "Shopping Master";
                shoppingStage = 5;
            }
            else if (GetBadgeData.shoppingLog >= 100)
            {
                a8.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingSix.png");
                a8Txt.Text = "Shopping Legend";
                shoppingStage = 6;
            }

            if (GetBadgeData.travelLog > 0 && GetBadgeData.travelLog < 5)
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelOne.png");
                a9Txt.Text = "Travel Novice";
                travelStage = 1;
            }
            else if (GetBadgeData.travelLog >= 5 && GetBadgeData.travelLog < 10)
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelTwo.png");
                a9Txt.Text = "Travel Apprentice";
                travelStage = 2;
            }
            else if (GetBadgeData.travelLog >= 10 && GetBadgeData.travelLog < 25)
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelThree.png");
                a9Txt.Text = "Travel Adept";
                travelStage = 3;
            }
            else if (GetBadgeData.travelLog >= 25 && GetBadgeData.travelLog < 50)
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFour.png");
                a9Txt.Text = "Travel Expert";
                travelStage = 4;
            }
            else if (GetBadgeData.travelLog >= 50 && GetBadgeData.travelLog < 100)
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFive.png");
                a9Txt.Text = "Travel Master";
                travelStage = 5;
            }
            else if (GetBadgeData.travelLog >= 100)
            {
                a9.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelSix.png");
                a9Txt.Text = "Travel Legend";
                travelStage = 6;
            }

            if (GetBadgeData.wasteLog > 0 && GetBadgeData.wasteLog < 5)
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteOne.png");
                a10Txt.Text = "Waste Novice";
                wasteStage = 1;
            }
            else if (GetBadgeData.wasteLog >= 5 && GetBadgeData.wasteLog < 10)
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteTwo.png");
                a10Txt.Text = "Waste Apprentice";
                wasteStage = 2;
            }
            else if (GetBadgeData.wasteLog >= 10 && GetBadgeData.wasteLog < 25)
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteThree.png");
                a10Txt.Text = "Waste Adept";
                wasteStage = 3;
            }
            else if (GetBadgeData.wasteLog >= 25 && GetBadgeData.wasteLog < 50)
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFour.png");
                a10Txt.Text = "Waste Expert";
                wasteStage = 4;
            }
            else if (GetBadgeData.wasteLog >= 50 && GetBadgeData.wasteLog < 100)
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFive.png");
                a10Txt.Text = "Waste Master";
                wasteStage = 5;
            }
            else if (GetBadgeData.wasteLog >= 100)
            {
                a10.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteSix.png");
                a10Txt.Text = "Waste Legend";
                wasteStage = 6;
            }

            if (GetBadgeData.waterLog > 0 && GetBadgeData.waterLog < 5)
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterOne.png");
                a11Txt.Text = "Water Novice";
                waterStage = 1;
            }
            else if (GetBadgeData.waterLog >= 5 && GetBadgeData.waterLog < 10)
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterTwo.png");
                a11Txt.Text = "Water Apprentice";
                waterStage = 2;
            }
            else if (GetBadgeData.waterLog >= 10 && GetBadgeData.waterLog < 25)
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterThree.png");
                a11Txt.Text = "Water Adept";
                waterStage = 3;
            }
            else if (GetBadgeData.waterLog >= 25 && GetBadgeData.waterLog < 50)
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFour.png");
                a11Txt.Text = "Water Expert";
                waterStage = 4;
            }
            else if (GetBadgeData.waterLog >= 50 && GetBadgeData.waterLog < 100)
```

```
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFive.png");
                a11Txt.Text = "Water Master";
                waterStage = 5;
            }
            else if (GetBadgeData.waterLog >= 100)
            {
                a11.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterSix.png");
                a11Txt.Text = "Water Legend";
                waterStage = 6;
            }

            if (GetBadgeData.workLog > 0 && GetBadgeData.workLog < 5)
            {
                a12.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workOne.png");
                a12Txt.Text = "Work Novice";
                workStage = 1;
            }
            else if (GetBadgeData.workLog >= 5 && GetBadgeData.workLog < 10)
            {
                a12.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workTwo.png");
                a12Txt.Text = "Work Apprentice";
                workStage = 2;
            }
            else if (GetBadgeData.workLog >= 10 && GetBadgeData.workLog < 25)
            {
                a12.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workThree.png");
                a12Txt.Text = "Work Adept";
                workStage = 3;
            }
            else if (GetBadgeData.workLog >= 25 && GetBadgeData.workLog < 50)
            {
                a12.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFour.png");
                a12Txt.Text = "Work Expert";
                workStage = 4;
            }
            else if (GetBadgeData.workLog >= 50 && GetBadgeData.workLog < 100)
            {
                a12.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFive.png");
                a12Txt.Text = "Work Master";
                workStage = 5;
            }
            else if (GetBadgeData.workLog >= 100)
            {
                a12.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workSix.png");
                a12Txt.Text = "Work Legend";
                workStage = 6;
            }
        }
        /** This function displays a popup when the first badge is tapped
        */
        private async void NavigateToBadgePopUpOne(object sender, EventArgs e)
        {
            int number = 1;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, habitsStage));
        }
        /** This function displays a popup when the second badge is tapped
        */
        private async void NavigateToBadgePopUpTwo(object sender, EventArgs e)
        {
            int number = 2;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, advancedStage));
        }
        /** This function displays a popup when the third badge is tapped
        */
        private async void NavigateToBadgePopUpThree(object sender, EventArgs e)
        {
            int number = 3;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, communityStage));
        }
        /** This function displays a popup when the fourth badge is tapped
        */
        private async void NavigateToBadgePopUpFour(object sender, EventArgs e)
        {
            int number = 4;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, energyStage));
        }
        /** This function displays a popup when the fifth badge is tapped
        */
        private async void NavigateToBadgePopUpFive(object sender, EventArgs e)
        {
            int number = 5;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, foodDrinkStage));
        }
        /** This function displays a popup when the sixth badge is tapped
        */
        private async void NavigateToBadgePopUpSix(object sender, EventArgs e)
        {
            int number = 6;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, homeStage));
        }
        /** This function displays a popup when the seventh badge is tapped
        */
        private async void NavigateToBadgePopUpSeven(object sender, EventArgs e)
        {
            int number = 7;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, outdoorsStage));
        }
        /** This function displays a popup when the eight badge is tapped
        */
        private async void NavigateToBadgePopUpEight(object sender, EventArgs e)
        {
            int number = 8;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, shoppingStage));
        }
        /** This function displays a popup when the ninth badge is tapped
        */
        private async void NavigateToBadgePopUpNine(object sender, EventArgs e)
        {
            int number = 9;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, travelStage));
        }
        /** This function displays a popup when the tenth badge is tapped
        */
        private async void NavigateToBadgePopUpTen(object sender, EventArgs e)
        {
            int number = 10;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, wasteStage));
        }
        /** This function displays a popup when the eleventh badge is tapped
        */
        private async void NavigateToBadgePopUpEleven(object sender, EventArgs e)
        {
            int number = 11;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, waterStage));
        }
        /** This function displays a popup when the twelfth badge is tapped
        */
        private async void NavigateToBadgePopUpTwelve(object sender, EventArgs e)
        {
            int number = 12;
            await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, workStage));
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.Views.ProfilePage.TopTabProfile"
             xmlns:local="clr-namespace:Application_Green_Quake.Models">
    <ContentPage.Content>
        <ScrollView>
            <StackLayout Spacing="0" Padding="0" BackgroundColor="#002a1e">
                <Frame Padding="0" CornerRadius="80"  Margin="120,10,120,10" BackgroundColor="#33554b" HasShadow="False">
                    <Frame Padding="0" CornerRadius="80" Margin="0,10,0,10" HeightRequest="90" WidthRequest="90" HorizontalOptions="CenterAndExpand" BackgroundColor="White">
                        <Image x:Name="ProfilePic" Source="{local:ImageResource Application_Green_Quake.Images.user.png}" Aspect="AspectFill">
                            <Image.GestureRecognizers>
                                <TapGestureRecognizer Tapped="ImageClicked"/>
                            </Image.GestureRecognizers>
                        </Image>
                    </Frame>
                </Frame>

                <StackLayout HeightRequest="100" VerticalOptions="Start" HorizontalOptions="FillAndExpand" Spacing="0" BackgroundColor="#002a1e">
                    <Label x:Name="Username" FontSize="28" HorizontalOptions="Center" VerticalOptions="Center" FontAttributes="Bold" TextColor="White"/>
                    <Entry x:Name="Bio" Text="Bio" Completed="SaveText" TextColor="White" BackgroundColor="#002a1e" HorizontalOptions="FillAndExpand" VerticalOptions="CenterAndExpand" Margin="40,0,40,0" HorizontalTextAlignment="Center" PlaceholderColor="White"/>
                </StackLayout>

                <StackLayout Orientation="Horizontal" HeightRequest="50" BackgroundColor="#002a1e" Padding="5">
                    <StackLayout Spacing="0" BackgroundColor="#002a1e" Orientation="Horizontal" HorizontalOptions="Start">
                        <Image Source="{local:ImageResource Application_Green_Quake.Images.TrophyCase.png}" WidthRequest="40" HeightRequest="40" HorizontalOptions="StartAndExpand" VerticalOptions="Center" Aspect="AspectFill">
                            <Image.GestureRecognizers>
                                <TapGestureRecognizer Tapped="NavigateToTrophyCase"/>
                            </Image.GestureRecognizers>
                        </Image>
                        <Label FontSize="14" TextColor="White" Text="Trophy Case" HorizontalOptions="StartAndExpand" VerticalOptions="Center">
                            <Label.GestureRecognizers>
                                <TapGestureRecognizer Tapped="NavigateToTrophyCase"/>
                            </Label.GestureRecognizers>
                        </Label>
                    </StackLayout>

                    <StackLayout Spacing="0" BackgroundColor="#002a1e" Orientation="Horizontal" HorizontalOptions="EndAndExpand">
                        <Image Source="{local:ImageResource Application_Green_Quake.Images.Achievements.png}" WidthRequest="40" HeightRequest="40" HorizontalOptions="StartAndExpand" VerticalOptions="Center" Aspect="AspectFill">
                            <Image.GestureRecognizers>
                                <TapGestureRecognizer Tapped="NavigateToAchievements"/>
                            </Image.GestureRecognizers>
                        </Image>
                        <Label FontSize="14" TextColor="White" Text="Achievements" HorizontalOptions="StartAndExpand" VerticalOptions="Center">
                            <Label.GestureRecognizers>
                                <TapGestureRecognizer Tapped="NavigateToAchievements"/>
                            </Label.GestureRecognizers>
                        </Label>
                    </StackLayout>

                    <StackLayout Spacing="0" BackgroundColor="#002a1e" Orientation="Horizontal" HorizontalOptions="EndAndExpand">
                        <Image Source="{local:ImageResource Application_Green_Quake.Images.Badges.png}" WidthRequest="40" HeightRequest="40" HorizontalOptions="StartAndExpand" VerticalOptions="Center" Aspect="AspectFill">
                            <Image.GestureRecognizers>
                                <TapGestureRecognizer Tapped="NavigateToBadges"/>
                            </Image.GestureRecognizers>
                        </Image>
                        <Label FontSize="14" TextColor="White" Text="Badges" HorizontalOptions="StartAndExpand" VerticalOptions="Center" >
                            <Label.GestureRecognizers>
                                <TapGestureRecognizer Tapped="NavigateToBadges"/>
                            </Label.GestureRecognizers>
                        </Label>
                    </StackLayout>
                </StackLayout>
                <StackLayout VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand" Spacing="0" BackgroundColor="#002a1e">
                    <Label x:Name="theLevel" TextColor="White" Text="Level" HorizontalOptions="Center" FontAttributes="Bold" FontSize="20"/>
                    <ProgressBar Margin="40,0,40,0" ProgressColor="Gold" x:Name="progressbar"/>
                    <Image x:Name="mosiac" Source="{local:ImageResource Application_Green_Quake.Images.Mosaics.lockedMosaics.jpg}" VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand" Aspect="Fill"/>
                </StackLayout>
            </StackLayout>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

```
/*! \class The TopTabProfile View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the TopTabProfile View Class. The progile page class.
 *
 */
using Application_Green_Quake.Models;
using Application_Green_Quake.ViewModels;
using Firebase.Database;
using Firebase.Database.Query;
using Firebase.Storage;
using Rg.Plugins.Popup.Services;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.ProfilePage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TopTabProfile : ContentPage
    {
        IAuth auth;
        float progress = 0;
        string bioInput = "";
        float count = 0;

        public TopTabProfile()
        {
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();
            OnAppearing();
        }
        /** This function is called before the page is displayed. It displays the images as
the criteria are met and also the bio and username. It also
         * calulates the players level and progress and displays the progress to the next
level on a progress bar.
        */
        protected override async void OnAppearing()
        {
            Username.Text = GetData.username;
            try
            {
                ProfilePic.Source = await new FirebaseStorage("application-green-
quake.appspot.com")
                .Child(auth.GetUid())
                .Child("Profile.jpg")
                .GetDownloadUrlAsync();
            }
            catch (Exception e)
            {
                Console.Write(e);
            }

            // Calculate the progress for the next level
            progress = (float)GetData.points / 10;
            progress = (int)(((decimal)progress % 1) * 10);

            progress = progress / 10;

            count = progress;
            count = count * 10;

            // Set the theLvl and animate the progress bar
            theLevel.Text =  "Lvl:  " + GetData.lvl.ToString() + " Points: " +
count.ToString() + " /10";
            await progressbar.ProgressTo(progress, 500, Easing.Linear);

            if (GetData.lvl == 1)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineOne.jpg");
            }
            else if (GetData.lvl == 2)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineTwo.jpg");
            }
            else if (GetData.lvl == 3)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineThree.jpg");
            }
            else if (GetData.lvl == 4)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineFour.jpg");
            }
            else if (GetData.lvl == 5)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1.jpg");
            }
            else if (GetData.lvl == 6)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineOne.jpg");
            }
            else if (GetData.lvl == 7)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineTwo.jpg");
            }
            else if (GetData.lvl == 8)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineThree.jpg");
            }
            else if (GetData.lvl == 9)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineFour.jpg");
            }
            else if (GetData.lvl == 10)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2.jpg");
            }
            else if (GetData.lvl == 11)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineOne.jpg");
```

```
            }
            else if (GetData.lvl == 12)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineTwo.jpg");
            }
            else if (GetData.lvl == 13)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineThree.jpg");
            }
            else if (GetData.lvl == 14)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineFour.jpg");
            }
            else if (GetData.lvl == 15)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3.jpg");
            }
            else if (GetData.lvl == 16)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineOne.jpg");
            }
            else if (GetData.lvl == 17)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineTwo.jpg");
            }
            else if (GetData.lvl == 18)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineThree.jpg");
            }
            else if (GetData.lvl == 19)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineFour.jpg");
            }
            else if (GetData.lvl == 20)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4.jpg");
            }
            else if (GetData.lvl == 21)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineOne.jpg");
            }
            else if (GetData.lvl == 22)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineTwo.jpg");
            }
            else if (GetData.lvl == 23)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineThree.jpg");
            }
            else if (GetData.lvl == 24)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineFour.jpg");
            }
            else if (GetData.lvl == 25)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5.jpg");
            }
            else if (GetData.lvl == 26)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineOne.jpg");
            }
            else if (GetData.lvl == 27)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineTwo.jpg");
            }
            else if (GetData.lvl == 28)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineThree.jpg");
            }
            else if (GetData.lvl == 29)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineFour.jpg");
            }
            else if (GetData.lvl == 30)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6.jpg");
            }
            else if (GetData.lvl == 31)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineOne.jpg");
            }
            else if (GetData.lvl == 32)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineTwo.jpg");
            }
            else if (GetData.lvl == 33)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineThree.jpg");
            }
            else if (GetData.lvl == 34)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineFour.jpg");
            }
            else if (GetData.lvl == 35)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7.jpg");
            }
            else if (GetData.lvl == 36)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineOne.jpg");
            }
            else if (GetData.lvl == 37)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineTwo.jpg");
            }
            else if (GetData.lvl == 38)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineThree.jpg");
            }
            else if (GetData.lvl == 39)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineFour.jpg");
            }
            else if (GetData.lvl == 40)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8.jpg");
            }
            else if (GetData.lvl == 41)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineOne.jpg");
            }
            else if (GetData.lvl == 42)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineTwo.jpg");
            }
            else if (GetData.lvl == 43)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineThree.jpg");
            }
            else if (GetData.lvl == 44)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineFour.jpg");
            }
            else if (GetData.lvl == 45)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9.jpg");
            }
            else if (GetData.lvl == 46)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineOne.jpg");
            }
            else if (GetData.lvl == 47)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineTwo.jpg");
            }
            else if (GetData.lvl == 48)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineThree.jpg");
            }
            else if (GetData.lvl == 49)
            {
                mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineFour.jpg");
            }
            else if (GetData.lvl == 50)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10.jpg");
            }
            else if (GetData.lvl == 51)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineOne.jpg");
            }
            else if (GetData.lvl == 52)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineTwo.jpg");
            }
            else if (GetData.lvl == 53)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineThree.jpg");
            }
            else if (GetData.lvl == 54)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineFour.jpg");
            }
            else if (GetData.lvl == 55)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11.jpg");
            }
            else if (GetData.lvl == 56)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineOne.jpg");
            }
            else if (GetData.lvl == 57)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineTwo.jpg");
            }
            else if (GetData.lvl == 58)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineThree.jpg");
            }
            else if (GetData.lvl == 59)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineFour.jpg");
            }
            else if (GetData.lvl == 60)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12.jpg");
            }
            else if (GetData.lvl == 61)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineOne.jpg");
            }
            else if (GetData.lvl == 62)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineTwo.jpg");
            }
            else if (GetData.lvl == 63)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineThree.jpg");
            }
            else if (GetData.lvl == 64)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineFour.jpg");
            }
            else if (GetData.lvl == 65)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13.jpg");
            }
            else if (GetData.lvl == 66)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineOne.jpg");
            }
            else if (GetData.lvl == 67)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineTwo.jpg");
            }
            else if (GetData.lvl == 68)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineThree.jpg");
            }
            else if (GetData.lvl == 69)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineFour.jpg");
            }
            else if (GetData.lvl == 70)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14.jpg");
            }
            else if (GetData.lvl == 71)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineOne.jpg");
            }
            else if (GetData.lvl == 72)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineTwo.jpg");
            }
            else if (GetData.lvl == 73)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineThree.jpg");
            }
            else if (GetData.lvl == 74)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineFour.jpg");
```

```
            }
            else if (GetData.lvl == 75)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15.jpg");
            }
            else if (GetData.lvl == 76)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineOne.jpg");
            }
            else if (GetData.lvl == 77)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineTwo.jpg");
            }
            else if (GetData.lvl == 78)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineThree.jpg");
            }
            else if (GetData.lvl == 79)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineFour.jpg");
            }
            else if (GetData.lvl == 80)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16.jpg");
            }
            else if (GetData.lvl == 81)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineOne.jpg");
            }
            else if (GetData.lvl == 82)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineTwo.jpg");
            }
            else if (GetData.lvl == 83)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineThree.jpg");
            }
            else if (GetData.lvl == 84)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineFour.jpg");
            }
            else if (GetData.lvl == 85)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17.jpg");
            }
            else if (GetData.lvl == 86)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineOne.jpg");
            }
            else if (GetData.lvl == 87)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineTwo.jpg");
            }
            else if (GetData.lvl == 88)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineThree.jpg");
            }
            else if (GetData.lvl == 89)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineFour.jpg");
            }
            else if (GetData.lvl == 90)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18.jpg");
            }
            else if (GetData.lvl == 91)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineOne.jpg");
            }
            else if (GetData.lvl == 92)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineTwo.jpg");
            }
            else if (GetData.lvl == 93)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineThree.jpg");
            }
            else if (GetData.lvl == 94)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineFour.jpg");
            }
            else if (GetData.lvl == 95)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19.jpg");
            }
            else if (GetData.lvl == 96)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineOne.jpg");
            }
            else if (GetData.lvl == 97)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineTwo.jpg");
            }
            else if (GetData.lvl == 98)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineThree.jpg");
            }
            else if (GetData.lvl == 99)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineFour.jpg");
            }
```

```
            else if (GetData.lvl == 100)
            {
                mosiac.Source = 
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20.jpg");
            }

            GetData data = new GetData();
            data.SetLvl();
        }
        /** This function navigates to UploadImagePopUp
        */
        private async void ImageClicked(object sender, EventArgs e)
        {
            await PopupNavigation.Instance.PushAsync(new UploadImagePopUp());
        }
        /** This function navigates to TrophyCase at tab 0
        */
        private async void NavigateToTrophyCase(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new TrophyCase(0));
        }
        /** This function navigates to TrophyCase at tab 1
        */
        private async void NavigateToAchievements(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new TrophyCase(1));
        }
        /** This function navigates to TrophyCase at tab 2
        */
        private async void NavigateToBadges(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new TrophyCase(2));
        }
        /** This function saves the users bio
       */
        private async void SaveText(object sender, EventArgs e)
        {
            FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");

            bioInput = Bio.Text;

            await firebaseClient
                    .Child("users")
                    .Child(auth.GetUid())
                    .PutAsync(new Users() {username = GetData.username ,bio = bioInput, 
nation = GetData.nation});
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:views="clr-namespace:Application_Green_Quake.Views" xmlns:views1="clr-namespace:Application_Green_Quake.Views.ProfilePage"
             xmlns:models="clr-namespace:Application_Green_Quake.Models;assembly=Application Green Quake"
             x:Class="Application_Green_Quake.Views.ProfilePage.TrophyCase"
            NavigationPage.HasBackButton="False"
            NavigationPage.HasNavigationBar="False">
    <ContentPage Title="Trophies">
        <ContentPage.Content>
            <ScrollView>
                <Grid HorizontalOptions="Fill" VerticalOptions="Fill" Padding="10,30,10,10" Row="20">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="*"/>
                        <RowDefinition Height="*"/>
                        <RowDefinition Height="*"/>
                        <RowDefinition Height="*"/>
                    </Grid.RowDefinitions>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*"/>
                    </Grid.ColumnDefinitions>
                    <StackLayout Grid.Row="0" >
                        <Image x:Name="t1"
                               Source="{models:ImageResource Application_Green_Quake.Images.lockOne.png}"
                               HorizontalOptions="CenterAndExpand">
                        </Image>
                        <Label x:Name="t1Txt" Text="1000 Points to unlock" TextColor="Black" HorizontalTextAlignment="Center"/>
                    </StackLayout>
                    <StackLayout Grid.Row="1">
                        <Image  x:Name="t2"
                                Source="{models:ImageResource Application_Green_Quake.Images.lockOne.png}"
                                HorizontalOptions="CenterAndExpand">
                        </Image>
                        <Label x:Name="t2Txt" Text="500 Points to unlock" TextColor="Black" HorizontalTextAlignment="Center"/>
                    </StackLayout>
                    <StackLayout Grid.Row="2">
                        <Image x:Name="t3"
                               Source="{models:ImageResource Application_Green_Quake.Images.lockOne.png}"
                               HorizontalOptions="CenterAndExpand">
                        </Image>
                        <Label x:Name="t3Txt" Text="250 Points to unlock" TextColor="Black" HorizontalTextAlignment="Center"/>
                    </StackLayout>
                    <StackLayout Grid.Row="3">
                        <Image x:Name="t4"
                               Source="{models:ImageResource Application_Green_Quake.Images.lockOne.png}"
                               HorizontalOptions="CenterAndExpand">
                        </Image>
                        <Label x:Name="t4Txt" Text="100 Points to unlock" TextColor="Black" HorizontalTextAlignment="Center"/>
                    </StackLayout>
                </Grid>
            </ScrollView>
        </ContentPage.Content>
    </ContentPage>

    <views1:Achievements Title="Achievements"></views1:Achievements>
    <views1:Badges Title="Badges"></views1:Badges>
</TabbedPage>
```

```
/*! \class The TrophyCase View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the TrophyCase View Class. This class is the class that displays all the trophies on the trophy page.
 *
 */
using Application_Green_Quake.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Application_Green_Quake.Views.ProfilePage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TrophyCase : TabbedPage
    {
        /** The constructor for Main menu
        @param tab supplied to tell the class which tabbed page to display.
        */
        public TrophyCase(int tab)
        {
            InitializeComponent();
            CurrentPage = Children[tab];
            OnAppearing();
        }
        /** This function is called before the page is displayed. It displays the image as the criteria is met
        */
        protected override void OnAppearing()
        {
            if (GetData.points >= 1000)
            {
                t1.Source = ImageSource.FromResource("Application_Green_Quake.Images.Trophies.diamond.png");
                t1Txt.Text = "Diamond Trophy";
            }
            if (GetData.points >= 500)
            {
                t2.Source = ImageSource.FromResource("Application_Green_Quake.Images.Trophies.gold.png");
                t2Txt.Text = "Gold Trophy";
            }
            if (GetData.points >= 250)
            {
                t3.Source = ImageSource.FromResource("Application_Green_Quake.Images.Trophies.silver.png");
                t3Txt.Text = "Silver Trophy";
            }
            if (GetData.points >= 100)
            {
                t4.Source = ImageSource.FromResource("Application_Green_Quake.Images.Trophies.bronze.png");
                t4Txt.Text = "Bronze Trophy";
            }
        }
    }
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pages:PopupPage x:Class="Application_Green_Quake.Views.ProfilePage.UploadImagePopUp"
                 xmlns="http://xamarin.com/schemas/2014/forms"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 xmlns:pages="clr-namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
                 BackgroundColor="#002a1e">

    <StackLayout HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" >
        <Image x:Name="ChosenImage" HeightRequest="200"></Image>
        <Button x:Name="Take" TextColor="White" BackgroundColor="#50C878" Text="Take Photo" Clicked="CapturePhotoClicked"></Button>
        <Button x:Name="Pick" TextColor="White" BackgroundColor="#50C878" Text="Pick From Storage" Clicked="UplaodPhotoClicked"></Button>
        <Button x:Name="Save" TextColor="White" BackgroundColor="#50C878" Text="Save" Clicked="storeImageClicked"></Button>
    </StackLayout>

</pages:PopupPage>
```

```
/*! \class The UploadImagePopUp View Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the UploadImagePopUp View Class. This class is the popup that appears to upload a new profile picture.
 *
 */
using Plugin.Media;
using System;
using System.Diagnostics;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.IO;
using System.Threading.Tasks;
using Acr.UserDialogs;
using Firebase.Storage;
using Plugin.Media.Abstractions;
using Rg.Plugins.Popup.Services;

namespace Application_Green_Quake.Views.ProfilePage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class UploadImagePopUp
    {
        IAuth auth;
        public MediaFile File { get; set; }

        public UploadImagePopUp()
        {
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();
        }
        /** This function enables the capturing of an image.
        */
        private async void CapturePhotoClicked(object sender, System.EventArgs e)
        {
            await CrossMedia.Current.Initialize();

            if (!CrossMedia.Current.IsCameraAvailable || !CrossMedia.Current.IsTakePhotoSupported)
            {
                await DisplayAlert("No Camera", "No Camera Detected", "OK");
                return;
            }

            File = await CrossMedia.Current.TakePhotoAsync(
                new StoreCameraMediaOptions
                {
                    SaveToAlbum = true
                }
                );
            if (File == null)
                return;

            ChosenImage.Source = ImageSource.FromStream(() =>
            {
                var stream = File.GetStream();
                return stream;
            });
        }

        /** This function enables the uploading of an image.
        */
        private async void UplaodPhotoClicked(object sender, System.EventArgs e)
        {
            await CrossMedia.Current.Initialize();
            try
            {
                File = await CrossMedia.Current.PickPhotoAsync(new PickMediaOptions
                {
                    PhotoSize = PhotoSize.Medium
                });
                if (File == null)
                    return;
                ChosenImage.Source = ImageSource.FromStream(() =>
                {
                    var imageSteram = File.GetStream();
                    return imageSteram;
                });
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex.Message);
            }
        }

        /** This function enables the storing of an image.
        */
        private async void storeImageClicked(object sender, System.EventArgs e)
        {
            UserDialogs.Instance.ShowLoading();
            try
            {
                {
                    await StoreImages(File.GetStream());
                }


                async Task<string> StoreImages(Stream imageStream)
                {
                    var stroageImage = await new FirebaseStorage("application-green-
quake.appspot.com")
                        .Child(auth.GetUid())
                        .Child("Profile.jpg")
                        .PutAsync(imageStream);
                    string imgurl = stroageImage;
                    return imgurl;
                }
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex.Message);

            }
            await Navigation.PushAsync(new MainMenu(2));
            await PopupNavigation.Instance.PopAsync(true);
            // Hide the loading screen
            UserDialogs.Instance.HideLoading();
        }
```

```
    }
}
```

```
/*! \class The IAuth Interface
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the IAuth Interface. It is the interface between the cross platform code and the native code.
 *
 */
using System;
using System.Threading.Tasks;

namespace Application_Green_Quake
{
    public interface IAuth
    {
        Task<String> LoginWithEmailAndPassword(string email, string password);

        Task<String> SignUpWithEmailAndPassword(string email, string password);

        Task ResetPassword(string email);

        bool SignOut();

        bool IsSignIn();

        string GetUid();
    }
}
```

```csharp
/*! \class The IAuth Interface
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the IAuth Interface. It is the interface between the cross platform code and the native code.
 *
 */
using System;
using System.Threading.Tasks;

namespace Application_Green_Quake
{
    public interface IAuth
    {
        Task<String> LoginWithEmailAndPassword(string email, string password);

        Task<String> SignUpWithEmailAndPassword(string email, string password);

        Task ResetPassword(string email);

        bool SignOut();

        bool IsSignIn();

        string GetUid();
    }
}
```

```
using Xamarin.Forms.Xaml;
[assembly: XamlCompilation(XamlCompilationOptions.Compile)]
namespace Application_Green_Quake
{

}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Application_Green_Quake.App">
    <Application.Resources>
        <!-- Colors -->
        <Color x:Key="AppPrimaryColor">Red</Color>
        <Color x:Key="AppBackgroundColor">White</Color>
        <Color x:Key="PrimaryColor">Black</Color>
        <Color x:Key="SecondaryColor">#50C878</Color>
        <Color x:Key="TertiaryColor">Yellow</Color>

        <!-- Implicit styles -->
        <Style TargetType="ContentPage"
               ApplyToDerivedTypes="True">
            <Setter Property="BackgroundColor"
,                    Value="{StaticResource AppBackgroundColor}" />

        </Style>

        <Style TargetType="TabbedPage"
               ApplyToDerivedTypes="True">
            <Setter Property="BarBackgroundColor"
,                    Value="{StaticResource SecondaryColor}" />
            <Setter Property="UnselectedTabColor"
,                    Value="{StaticResource PrimaryColor}" />
            <Setter Property="SelectedTabColor"
,                    Value="{StaticResource AppBackgroundColor}" />
        </Style>

        <Style TargetType="NavigationPage"
               ApplyToDerivedTypes="True">
            <Setter Property="BarBackgroundColor"
,                    Value="{StaticResource SecondaryColor}"/>
        </Style>
    </Application.Resources>
</Application>
```

```
/*! \mainpage The App Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c002289
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the App Class. This is the class that gets loaded first on app launch.
 *
 */
using Application_Green_Quake.ViewModels;
using Microsoft.AppCenter;
using Microsoft.AppCenter.Analytics;
using Microsoft.AppCenter.Crashes;
using Application_Green_Quake.Views;
using Xamarin.Forms;

namespace Application_Green_Quake
{
    public partial class App : Application
    {
        IAuth auth;
        public App()
        {
            //Register Syncfusion license
Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense("NDM1OTg4QDMxMzkyZTMxMmUzME5wen
            GetData set = new GetData();
            set.SetLvl();
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();

            //If the user is signed in navigate to the main menu
            if (auth.IsSignIn())
            {
                MainPage = new NavigationPage(new MainMenu());
            }
            //If the users is not signed in navigate to the login screen
            else
            {
                MainPage = new NavigationPage(new MainPage());
            }
        }

        protected override void OnStart()
        {
            GetData data = new GetData();
            data.SetLvl();
            AppCenter.Start("android=87250b90-3ea3-429d-ac0b-7e47e6cd70ac;" +
                  "uwp={Your UWP App secret here};" +
                  "ios={Your iOS App secret here}",
                  typeof(Analytics), typeof(Crashes));
        }

        protected override void OnSleep()
        {
        }

        protected override void OnResume()
        {
            GetData data = new GetData();
            data.SetLvl();
        }
    }
}
```

```
/*! \class The AuthDroid Native Android Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the AuthDroid Native Android Class. This is implements native functions to check if a user is signed in, o try and log in, to try and
 * sign up to get their uid and to get a password rest email.
 *
 */
using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Application_Green_Quake.Droid;
using Firebase.Auth;

[assembly: Dependency(typeof(AuthDroid))]
namespace Application_Green_Quake.Droid
{
    public class AuthDroid : IAuth
    {
        /** This function checks if a user is signed in.
        */
        public bool IsSignIn()
        {
            var user = FirebaseAuth.Instance.CurrentUser;
            return user != null;

        }
        /** This function logs a user into the application
        */
        public async Task<string> LoginWithEmailAndPassword(string email, string password)
        {
            try
            {
                var user = await FirebaseAuth.Instance.SignInWithEmailAndPasswordAsync(email, password);
                var token = user.User.Uid;
                return token;
            }
            catch (FirebaseAuthInvalidUserException e)
            {
                e.PrintStackTrace();
                return string.Empty;
            }
            catch (FirebaseAuthInvalidCredentialsException e)
            {
                e.PrintStackTrace();
                return string.Empty;
            }
        }
        /** This function signs a user out of the application.
        */
        public bool SignOut()
        {
            try
            {
                FirebaseAuth.Instance.SignOut();
                return true;
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
                return false;
            }
        }
        /** This function allows a user to sign up into the app.
        */
        public async Task<string> SignUpWithEmailAndPassword(string email, string password)
        {
            try
            {
                var newUser = await
FirebaseAuth.Instance.CreateUserWithEmailAndPasswordAsync(email, password);
                var token = newUser.User.Uid;
                return token;
            }
            catch (FirebaseAuthUserCollisionException e)
            {
                e.PrintStackTrace();
                return "duplicate";
            }
            catch (FirebaseAuthInvalidUserException e)
            {
                e.PrintStackTrace();
                return string.Empty;
            }
            catch (FirebaseAuthInvalidCredentialsException e)
            {
                e.PrintStackTrace();
                return string.Empty;
            }
        }
        /** This function gets the currently signed in users UID.
        */
        public string GetUid()
        {
            var user = FirebaseAuth.Instance.CurrentUser;
            if (user != null)
            {
                try
                {
                    var uid = user.Uid;
                    return uid;
                }
                catch (FirebaseAuthInvalidUserException e)
                {
                    e.PrintStackTrace();
                    return string.Empty;
                }
                catch (FirebaseAuthInvalidCredentialsException e)
                {
                    e.PrintStackTrace();
                    return string.Empty;
                }
            }
            else
            {
                return "";
            }
        }
        /** This function sends the password reset email.
        */
        public async Task ResetPassword(string email)
        {
            await FirebaseAuth.Instance.SendPasswordResetEmailAsync(email);
        }
    }
}
```

```json
{
  "project_info": {
    "project_number": "637452254914",
    "firebase_url": "https://application-green-quake-default-rtdb.firebaseio.com",
    "project_id": "application-green-quake",
    "storage_bucket": "application-green-quake.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:637452254914:android:bc2c95ec69db3a6528455b",
        "android_client_info": {
          "package_name": "com.ApplicationGreenQuake"
        }
      },
      "oauth_client": [
        {
          "client_id": "637452254914-
4t61he1vvfrkq3l2d2c65410noqs06b1.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "AIzaSyBm_gRAhnZ6wVQ_j_sn9CNJ6J1aUq8toFw"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": [
            {
              "client_id": "637452254914-
4t61he1vvfrkq3l2d2c65410noqs06b1.apps.googleusercontent.com",
              "client_type": 3
            }
          ]
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

```
/*! \class The MainActivity Native Android Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the MainActivity Native Android Class. It had to be modified to make certain Nuget Packages and APIs work
 *
 */
using Acr.UserDialogs;
using Android.App;
using Android.Content.PM;
using Android.Runtime;
using Android.OS;
using Firebase;
using Plugin.CurrentActivity;

namespace Application_Green_Quake.Droid
{
    [Activity(Label = "Application_Green_Quake", Icon = "@drawable/AppLogo", Theme = "@style/MainTheme", MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize )]
    public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        /** This function makes it that when outside of a pop up is touched the popup closes.
        */
        public override void OnBackPressed()
        {
            Rg.Plugins.Popup.Popup.SendBackPressed(base.OnBackPressed);
        }
        protected override void OnCreate(Bundle savedInstanceState)
        {
            TabLayoutResource = Resource.Layout.Tabbar;
            ToolbarResource = Resource.Layout.Toolbar;

            base.OnCreate(savedInstanceState);

            UserDialogs.Init(this);
            Rg.Plugins.Popup.Popup.Init(this);
            CrossCurrentActivity.Current.Init(this, savedInstanceState);
            FirebaseApp.InitializeApp(Application.Context);
            Xamarin.Essentials.Platform.Init(this, savedInstanceState);
            Xamarin.Forms.Forms.Init(this, savedInstanceState);
            Xamarin.FormsGoogleMaps.Init(this, savedInstanceState); // initialize for Xamarin.Forms.GoogleMaps
            LoadApplication(new App());
        }

        public override void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
        {
            Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);
            base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }
}
```

```
/*! \class The MainApplication Native Android Class
 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c00228956@itcarlow.ie
 * \date 28/04/2021
 * \section desc_sec Description
 *
 * Description: This is the MainApplication Native Android Class. It had to be modified to make the APKs work
 *
 */
#if DEBUG
using System;
using Android.App;
using Android.Runtime;
using Application_Green_Quake.Models;
using Plugin.CurrentActivity;

[Application(Debuggable = true)]
#else
using Android.App;
using Android.Runtime;
using Application_Green_Quake.Models;
using Plugin.CurrentActivity;
using System;

[Application(Debuggable = false)]
#endif
[MetaData("com.google.android.maps.v2.API_KEY",
             Value = AppConstants.googleMapsApiKey)]
public class MainApplication : Application
{
    public MainApplication(IntPtr handle, JniHandleOwnership transer)
      : base(handle, transer)
    {
    }

    public override void OnCreate()
    {
        base.OnCreate();
        CrossCurrentActivity.Current.Init(this);
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1"
android:versionName="1.0" package="com.ApplicationGreenQuake"
android:installLocation="preferExternal">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="30" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application android:label="Application_Green_Quake.Android"
android:theme="@style/MainTheme">
        <uses-library android:name="org.apache.http.legacy" android:required="false" />
        <provider android:name="android.support.v4.content.FileProvider"
android:authorities="com.ApplicationGreenQuake.fileprovider" android:exported="false"
android:grantUriPermissions="true">
            <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
android:resource="@xml/file_paths"></meta-data>
        </provider>
    </application>
</manifest>
```