



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/1MIPChFA\\_Z\\_XOzcy3yi\\_Q1Gt6JyNJcC1e/view?usp=sharing](https://drive.google.com/file/d/1MIPChFA_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

<b>1 Introduction</b>	<b>3</b>
<b>2 Documentation</b>	<b>3</b>
<b>3 Installation</b>	<b>3</b>
<b>4 Application Screenshots</b>	<b>4</b>
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 H2O 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
<b>5 Database</b>	<b>17</b>
5.1 Firebase Database	18
5.2 Firebase Storage	21
5.3 Firebase Authentication	22
<b>6 Code Listings</b>	<b>22</b>

## 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.

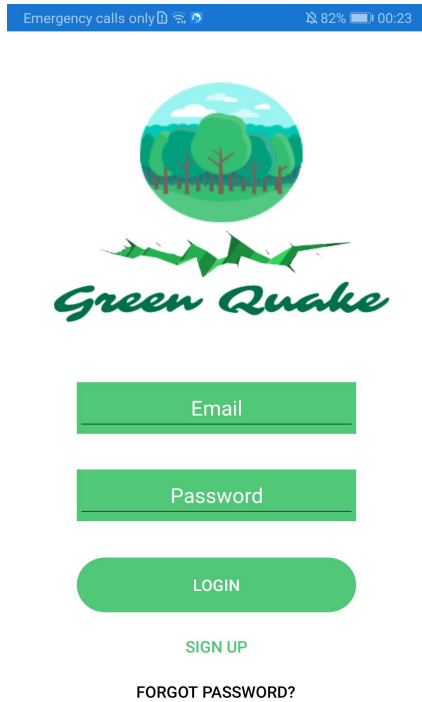
The second and more simpler way to install this application is to just simply download the APK that I have provided in the project folder on your device and just simply clicking on it and installing it.



## 4 Application Screenshots

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

### 4.1 Login Screen

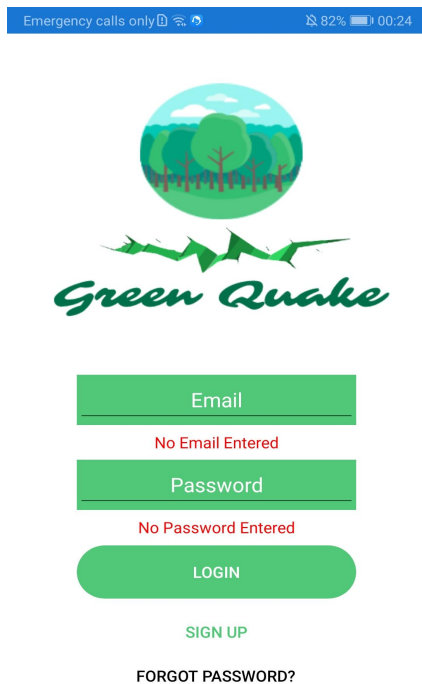


As you can see this is the **login** screen. This will be the opening screen for the application when the user is not signed in. A user can enter their email and password and click the login button to attempt to sign in or they can tap the Sign up option to navigate to the sign up page or they can tap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

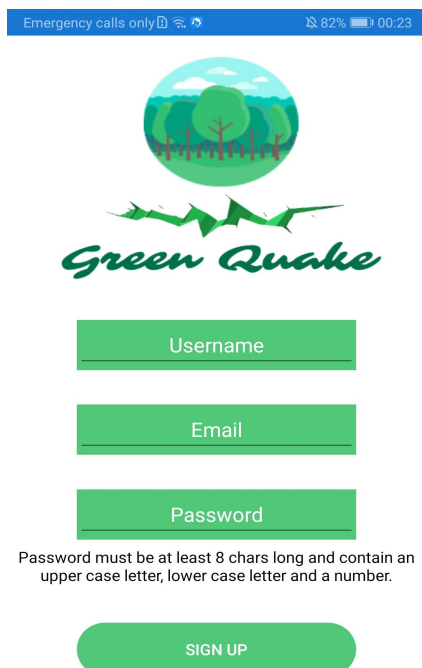
## 4.2 Login Screen With Error Messages



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.

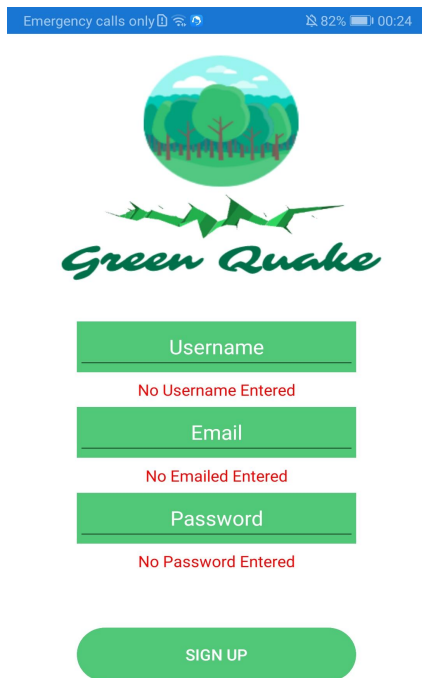
## 4.3 Sign Up Screen



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 up saying your account has been created and redirects the user to the login page.

**Figure 3:** Sign Up Screen.

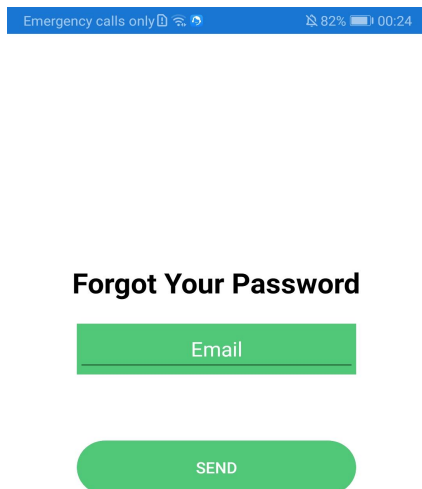
## 4.4 Sign Up Screen With Error Messages



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.

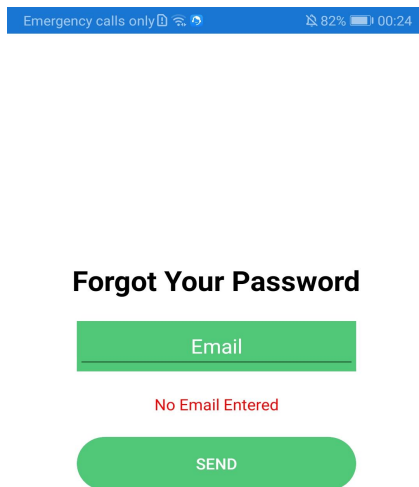
## 4.5 Forgot Password Screen



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.

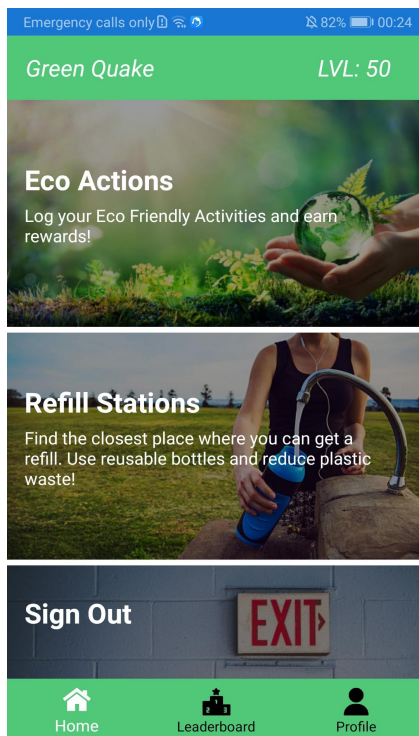
## 4.6 Forgot Password Screen With Error Messages



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.

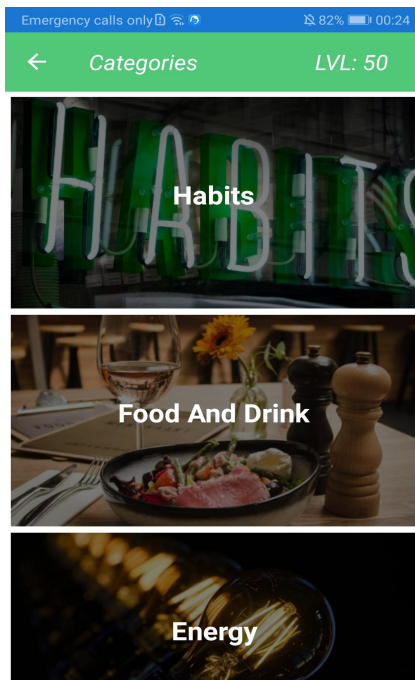
## 4.7 Main Menu Screen



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.

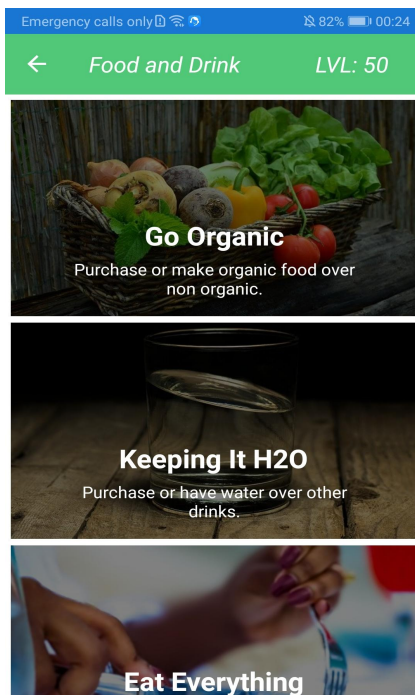
## 4.8 Categories Screen



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.

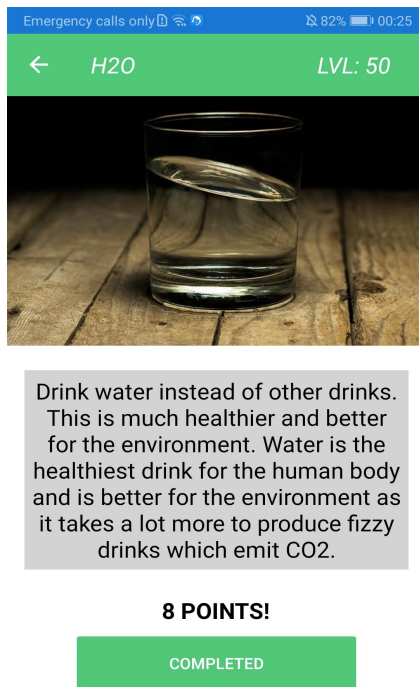
## 4.9 Food and Drink Subcategories Screen



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 H2O Action Screen

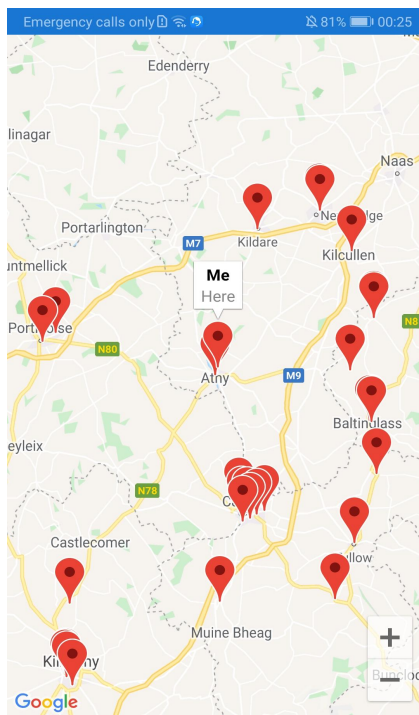


This is the **H2O** action screen. This screen provides some details and interesting information about this particular action. The amount of points that it is worth is displayed below the text. The user can press the Completed button to log the action. Once this is pressed either the points get posted 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: H2O Action Screen.

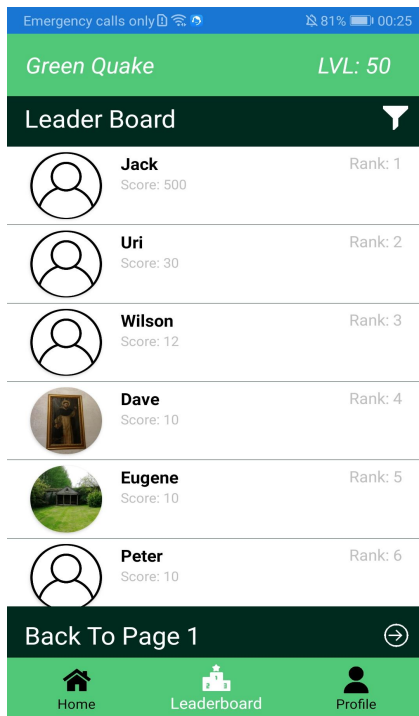
## 4.11 Refill Stations Screen



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.

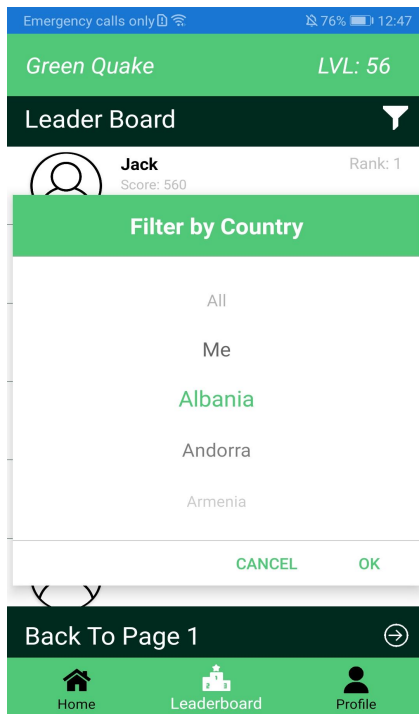
## 4.12 Leaderboard Screen



This is the **Leaderboard** screen that appears after a user selects the Leaderboard 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 load then an alert box with the appropriate message gets displayed.

Figure 12: Leaderboard Screen.

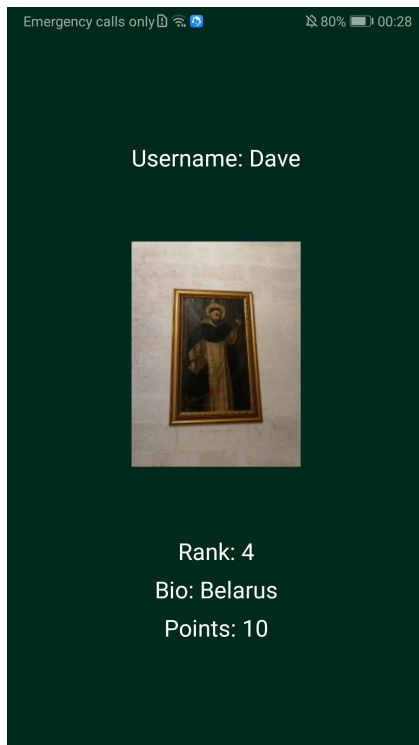
## 4.13 Leaderboard Screen When The Filter Button Is Tapped



This is the **Leaderboard** 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.

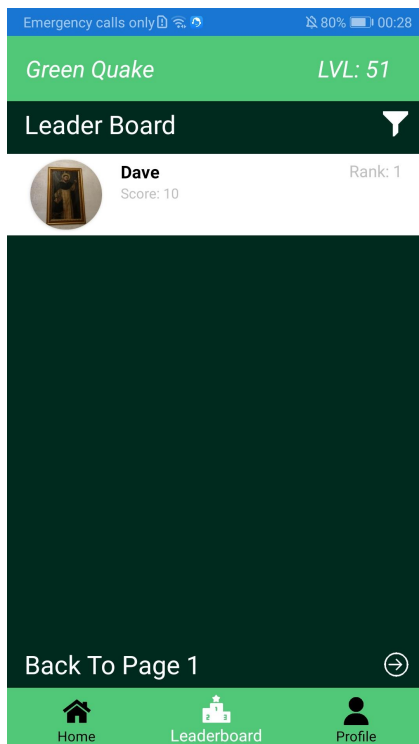
## 4.14 Leaderboard Pop Up Screen When A Profile Is Tapped On The Leaderboard



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

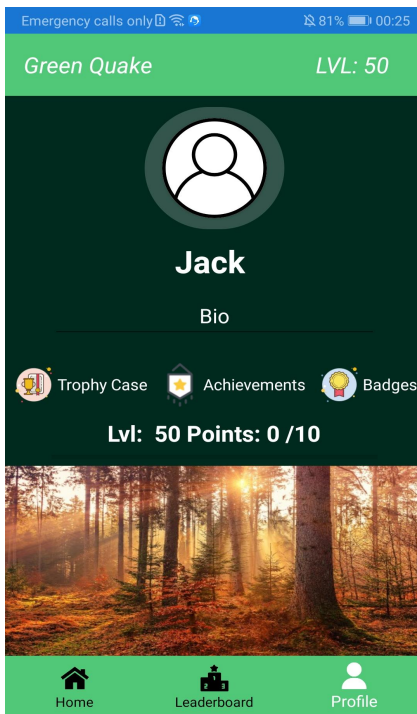


This is the **Leaderboard** after it has been **Filtered by Nation**. Only the profiles with the correct nation are displayed and they are ranked against other accounts of the same nation.

**Figure 22:** Filtered Leaderboard Screen.



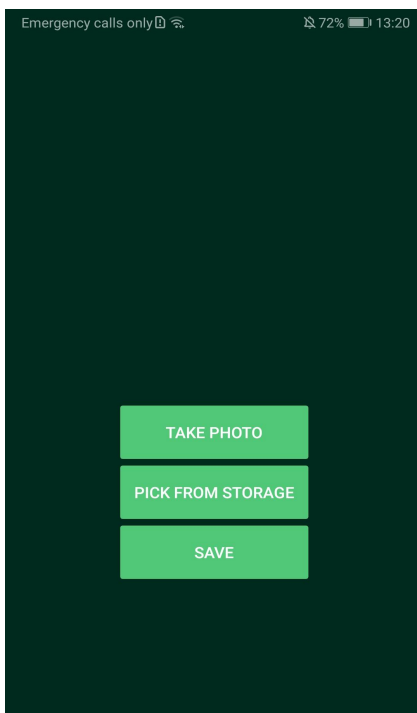
## 4.16 Profile Screen



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.

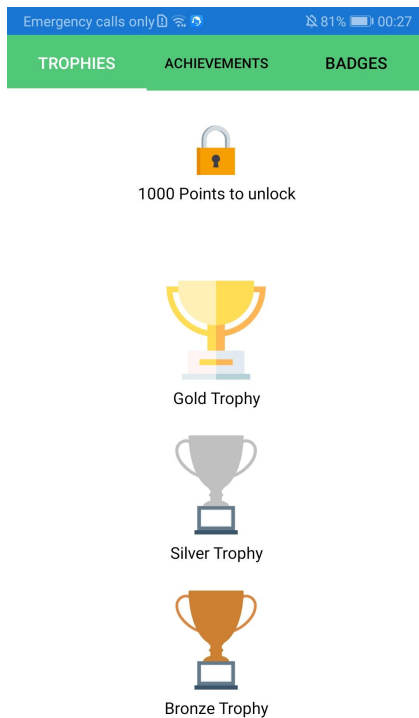
## 4.17 Popup Screen After The Profile Image Is Tapped



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.

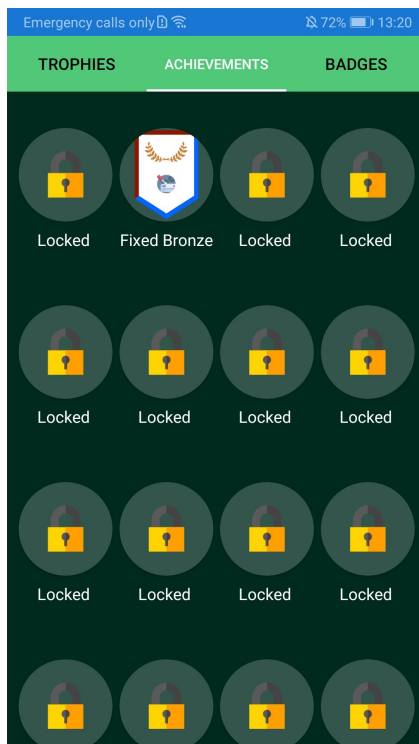
## 4.18 The Trophies Screen



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.

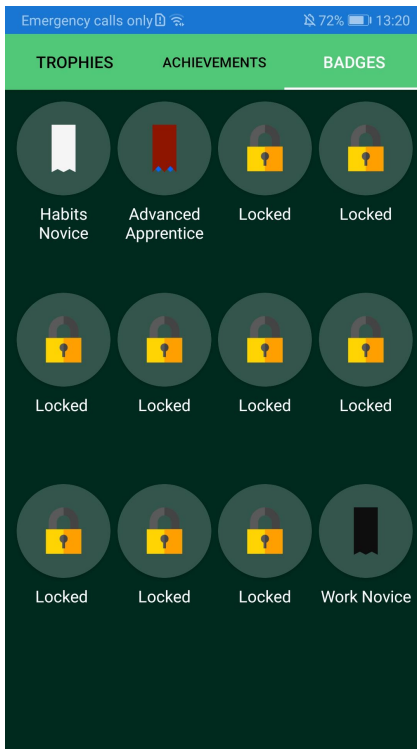
## 4.19 The Achievements Screen



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.

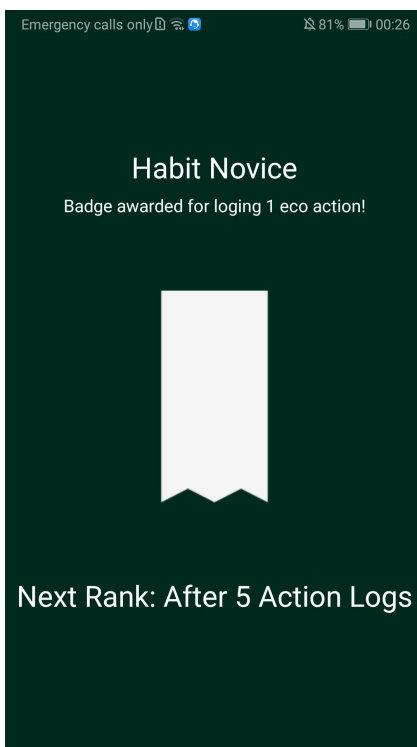
## 4.20 The BadgesScreen



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.

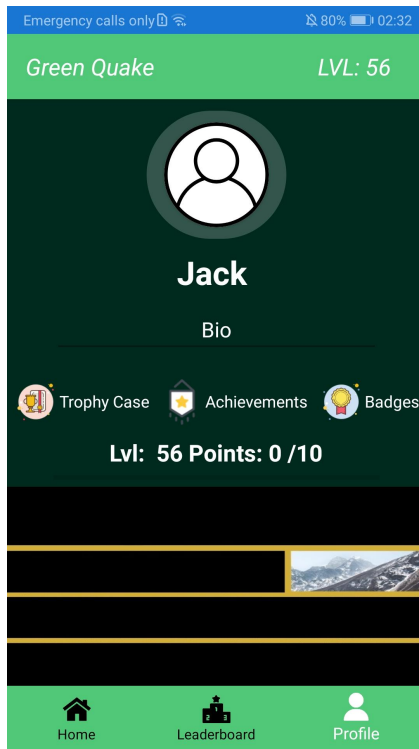
## 4.21 The Badge Popup Screen



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.

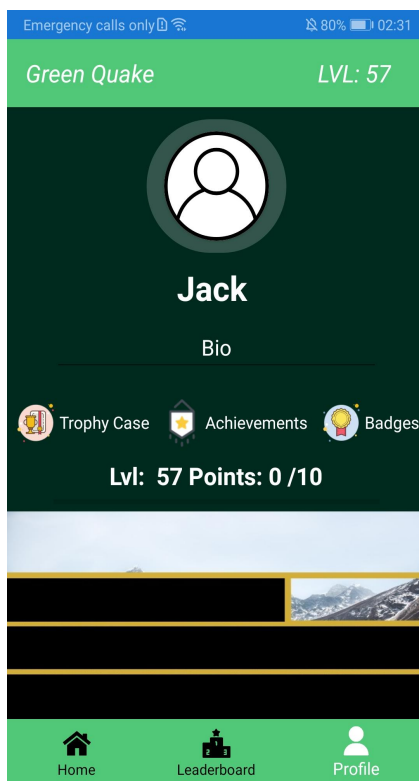
## 4.22 The Live Avatar Mosaics on Lvl 56



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

Figure 22: The Live Avatar Mosaics on Lvl 56.

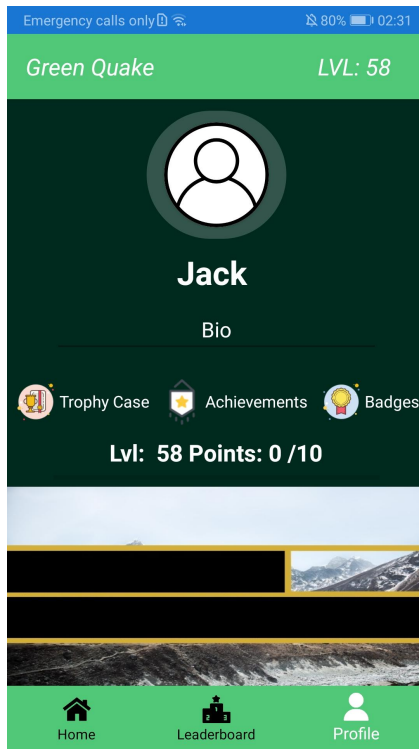
## 4.23 The Live Avatar Mosaics on Lvl 57



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

Figure 23: The Live Avatar Mosaics on Lvl 57.

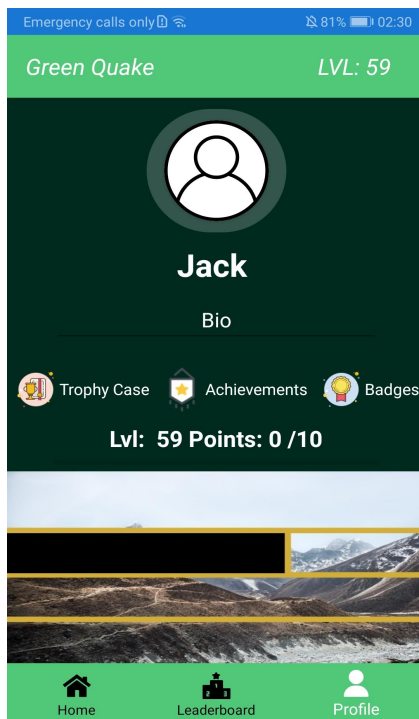
## 4.24 The Live Avatar Mosaics on Lvl 58



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

Figure 24: The Live Avatar Mosaics on Lvl 58.

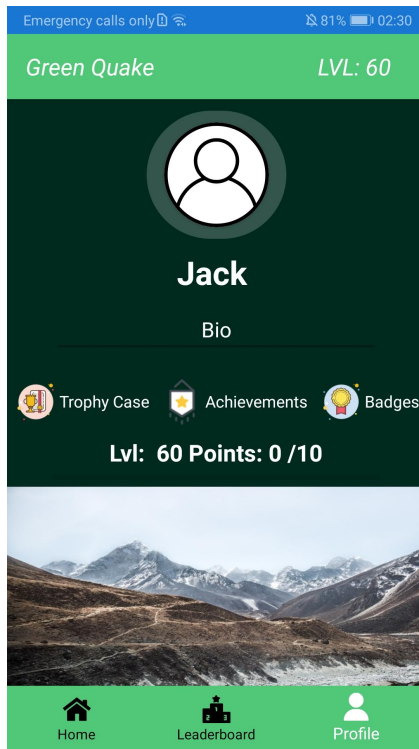
## 4.25 The Live Avatar Mosaics on Lvl 59



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

-|offLightsCount

-|points

-|showerCount

-|timedShowerCount

-|username



```
-|Points
  -|UID
    -|points
    -|username
-|SecurityChecks
  -|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 its structure can be seen below:

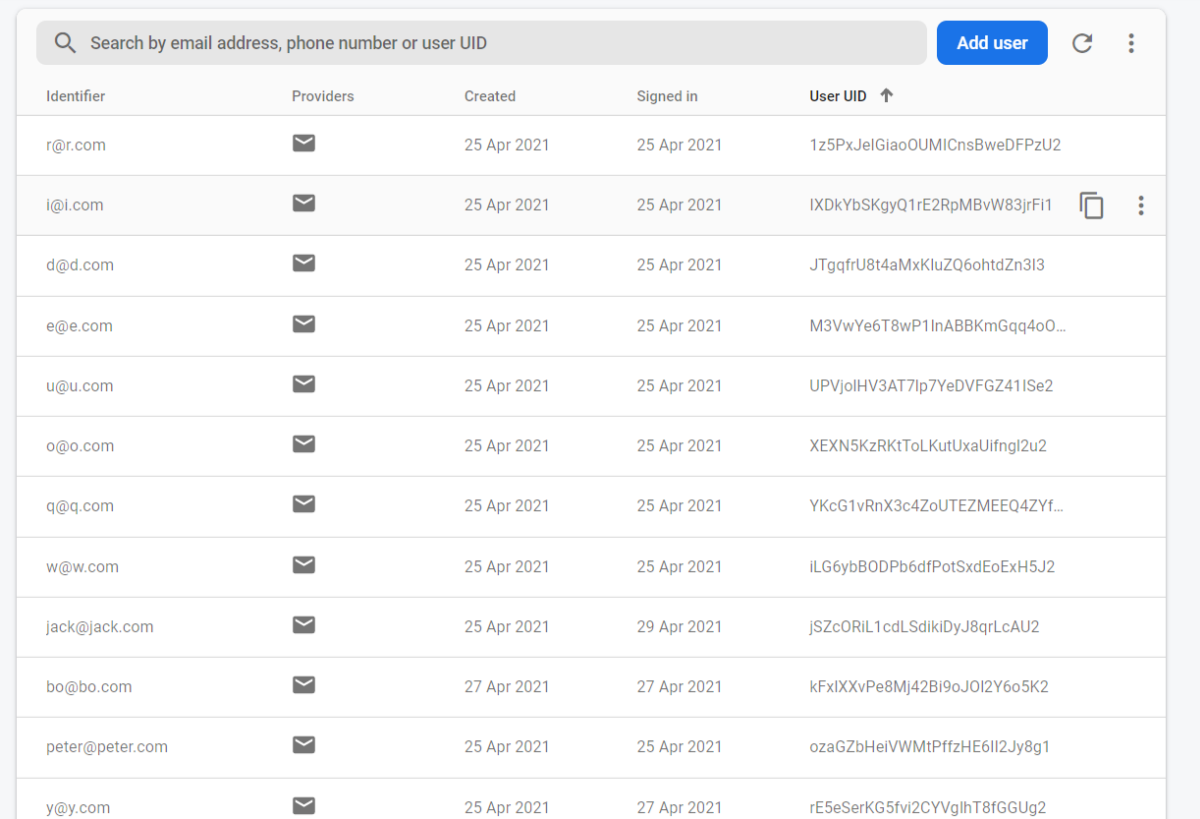
**-|gs://application-green-quake.appspot.com**

-|UID

-|filename

## 5.3 Firebase Authentication

The Firebase Authentication Database is shown below.



The screenshot shows the Firebase Authentication console interface. At the top, there is a search bar with the placeholder text "Search by email address, phone number or user UID". To the right of the search bar are three buttons: "Add user" (in blue), a refresh icon, and a menu icon. Below the search bar is a table with the following columns: "Identifier", "Providers", "Created", "Signed in", and "User UID" (with an upward arrow icon). The table contains 13 rows of user data. Each row has a copy icon and a menu icon on the right side.

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	IXDkYbSKgyQ1rE2RpMBvW83jrF1
d@d.com	✉	25 Apr 2021	25 Apr 2021	JTgqfrU8t4aMxKluZQ6ohtdZn3I3
e@e.com	✉	25 Apr 2021	25 Apr 2021	M3VwYe6T8wP1InABBKmGqq4o0...
u@u.com	✉	25 Apr 2021	25 Apr 2021	UPVjoiHV3AT7Ip7YeDVFZ41ISe2
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	ozaGZbHeiVWMtPffzHE6II2Jy8g1
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.

```
1 /!* \class The AdvancedPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the AdvancedPoints Model Class
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class AdvancedPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int fixCount { get; set; }
17     }
18 }
```

```
1  /*! \class The AppConstants Model Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the AppConstants Model Class in contains the constants that are used
   throughout this application.
7  *
8  */
9  namespace Application_Green_Quake.Models
10 {
11     public class AppConstants
12     {
13         //Int Constants
14         public const int twoPoints = 2;
15         public const int fourPoints = 4;
16         public const int sixPoints = 6;
17         public const int eightPoints = 8;
18         public const int tenPoints = 10;
19
20         //String Constants
21         public const string googleMapsApiKey = "AIzaSyDf7Bq7gjei8Sp1AS_SweapWyHe2rJtLmw";
22         public const string twoPointsMsg = "2 Points Point Have been added";
23         public const string fourPointsMsg = "4 Points Point Have been added";
24         public const string sixPointsMsg = "6 Points Point Have been added";
25         public const string eightPointsMsg = "8 Points Point Have been added";
26         public const string tenPointsMsg = "10 Points Point Have been added";
27     }
28 }
```

```
1 /!* \class The CommunityPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the CommunityPoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class CommunityPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int createGroupCount { get; set; }
17         public int communityCount { get; set; }
18         public int donateCount { get; set; }
19         public int groupCount { get; set; }
20         public int shareCount { get; set; }
21         public int awarenessCount { get; set; }
22     }
23 }
24 }
```

```
1 /!* \class The EnergyPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the EnergyPoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class EnergyPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int hangDryCount { get; set; }
17         public int fullDryerCount { get; set; }
18         public int insulateWaterCount { get; set; }
19         public int efficientThermostatCount { get; set; }
20         public int isolateHomeCount { get; set; }
21         public int ledLightBulbCount { get; set; }
22         public int fullMachineCount { get; set; }
23         public int microwaveCount { get; set; }
24         public int offSocketCount { get; set; }
25         public int reBatteriesCount { get; set; }
26         public int fridgeCount { get; set; }
27         public int draftSealCount { get; set; }
28         public int ductSealCount { get; set; }
29         public int solarPanelCount { get; set; }
30     }
31 }
```

```
1  /*! \class The FoodAndDrinkPoints Model Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the FoodAndDrinkPoints Model Class.
7  *
8  */
9  namespace Application_Green_Quake.Models
10 {
11     class FoodAndDrinkPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int organicCount { get; set; }
17         public int eatAllCount { get; set; }
18         public int foodDeliverCount { get; set; }
19         public int noMeatCount { get; set; }
20         public int ownCoffeeCount { get; set; }
21         public int reCoffeeMugCount { get; set; }
22         public int saveLeftOversCount { get; set; }
23         public int steelStrawCount { get; set; }
24         public int waterOverFizzyCount { get; set; }
25     }
26 }
```



```
1 /!* \class The HabitsPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the HabitsPoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class HabitsPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int brushingCount { get; set; }
17         public int fullWasherCount { get; set; }
18         public int showerCount { get; set; }
19         public int timedShowerCount { get; set; }
20         public int offLigtsCount { get; set; }
21         public int matchesCount { get; set; }
22     }
23 }
```

```
1 /!* \class The HomePoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the HomePoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class HomePoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int airOutCount { get; set; }
17         public int nonHarmCount { get; set; }
18         public int outsideCount { get; set; }
19         public int plantIntoHomeCount { get; set; }
20         public int toiletFlushCount { get; set; }
21     }
22 }
```

```
1  /*! \class The ImageResourceExtension Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ImageResourceExtension Class. It is used for loading image
   files.
7  *
8  */
9  using System;
10 using System.Reflection;
11 using Xamarin.Forms;
12 using Xamarin.Forms.Xaml;
13
14 namespace Application_Green_Quake.Models
15 {
16     [ContentProperty (nameof(Source))]
17     class ImageResourceExtension : IMarkupExtension
18     {
19         public string Source { get; set; }
20
21         public object ProvideValue(IServiceProvider serviceProvider)
22         {
23             //If the source is nothing then just null is returned
24             if (Source == null)
25             {
26                 return null;
27             }
28
29             var imageSource = ImageSource.FromResource(Source,
   typeof(ImageResourceExtension).GetTypeInfo().Assembly);
30
31             return imageSource;
32         }
33     }
34 }
35 }
```

```
1 /!* \class The LeaderBoard Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the LeaderBoard Model Class.
7 *
8 */
9 using Xamarin.Forms;
10
11 namespace Application_Green_Quake.Models
12 {
13     class LeaderBoard
14     {
15         public ImageSource image { get; set; }
16         public string username { get; set; }
17         public int points { get; set; }
18         public string rank { get; set; }
19         public string bio { get; set; }
20         public string nation { get; set; }
21         public string uid { get; set; }
22     }
23 }
```

```
1 /!* \class The OutdoorsPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the OutdoorsPoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class OutdoorsPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int campingCount { get; set; }
17         public int picnicCount { get; set; }
18         public int plantBushCount { get; set; }
19         public int plantFlowerCount { get; set; }
20         public int plantTreeCount { get; set; }
21         public int scoopCount { get; set; }
22         public int fruitGardenCount { get; set; }
23         public int herbGardenCount { get; set; }
24         public int vegetableGardenCount { get; set; }
25         public int birdFeederCount { get; set; }
26     }
27 }
28 }
```

```
1 / *! \class The Points Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
  c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the Points Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class Points
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15     }
16 }
17 }
```

```
1 /!* \class The SecurityChecks Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the SecurityChecks Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class SecurityChecks
12     {
13         public string date { get; set; }
14         public long time { get; set; }
15         public int counter { get; set; }
16     }
17 }
```

```
1 /!* \class The ShoppingPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the ShoppingPoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class ShoppingPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int clothNapkinCount { get; set; }
17         public int clothTowelCount { get; set; }
18         public int applianceCount { get; set; }
19         public int productCount { get; set; }
20         public int toothbrushCount { get; set; }
21         public int clothesCount { get; set; }
22         public int foodCount { get; set; }
23         public int localCount { get; set; }
24         public int looseLeafCount { get; set; }
25         public int organicFoodCount { get; set; }
26         public int reusableCount { get; set; }
27         public int reBatCount { get; set; }
28         public int reBagCount { get; set; }
29     }
30 }
```



```
1 /!* \class The Station Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the Station Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class Station
12     {
13         public string description { get; set; }
14
15         public string label { get; set; }
16
17         public double latitude { get; set; }
18
19         public double longitude { get; set; }
20     }
21 }
```

```
1 /!* \class The TravelPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the TravelPoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class TravelPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int carpoolCount { get; set; }
17         public int cycleCount { get; set; }
18         public int ecoCarCount { get; set; }
19         public int transportCount { get; set; }
20         public int walkCount { get; set; }
21     }
22 }
```

```
1 /!* \class The Usernames Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the Usernames Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class Usernames
12     {
13         public string Uid { get; set ; }
14     }
15 }
```

```
1 /!* \class The Users Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the Users Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class Users
12     {
13         public string username { get; set; }
14         public string bio { get; set; }
15         public string nation { get; set; }
16     }
17 }
```

```
1 /!* \class The WastePoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the WastePoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class WastePoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int billsCount { get; set; }
17         public int compostCount { get; set; }
18         public int setUpRecyclingBinCount { get; set; }
19         public int bioBinBagsCount { get; set; }
20         public int recyclingBinCount { get; set; }
21     }
22 }
```

```
1 /!* \class The WaterPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the WaterPoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class WaterPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int cisternCount { get; set; }
17         public int rainBarrelCount { get; set; }
18         public int reWaterCount { get; set; }
19         public int showerBucketCount { get; set; }
20         public int wSShowerHeadCount { get; set; }
21     }
22 }
```

```
1 /!* \class The WorkPoints Model Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the WorkPoints Model Class.
7 *
8 */
9 namespace Application_Green_Quake.Models
10 {
11     class WorkPoints
12     {
13         public string username { get; set; }
14         public int points { get; set; }
15         public int numberOfLogs { get; set; }
16         public int paperCount { get; set; }
17         public int offElectronicsCount { get; set; }
18         public int remoteWorkCount { get; set; }
19     }
20 }
```

```
1  /*! \class The AdvancedPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the AdvancedPointsUpdate ViewModel Class. It updates the data for
   the Advanced Category of the application. The functions in this class
7  * work by reading in all the chosen data and updating the selected fields and then sending
   this data to back firebase.
8  *
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class AdvancedPointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int fixCount2 = 0;
23         string username = "";
24
25         IAuth auth;
26         /** This function updates the points in the Advanced category by ten points. It
   also increments the number of logs logged in the Advanced
27         * category by one and increments the number of times this particular action was
   logged by one and sends this data to Firebase.
28         */
29         public async void FixPoints()
30         {
31             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtdb.firebaseio.com/");
32             auth = DependencyService.Get<IAuth>();
33             try
34             {
35                 username = (await firebaseClient
36                     .Child("users")
37                     .Child(auth.GetUid())
38                     .OnceSingleAsync<Users>()).username;
39
40                 points2 = (await firebaseClient
41                     .Child("AdvancedPoints")
42                     .Child(auth.GetUid())
43                     .OnceSingleAsync<AdvancedPoints>()).points;
44
45                 points2 = points2 + AppConstants.tenPoints;
46
47                 numberOfLogs2 = (await firebaseClient
48                     .Child("AdvancedPoints")
49                     .Child(auth.GetUid())
50                     .OnceSingleAsync<AdvancedPoints>()).numberOfLogs;
51
52                 numberOfLogs2++;
53
54                 fixCount2 = (await firebaseClient
55                     .Child("AdvancedPoints")
56                     .Child(auth.GetUid())
```



```
57         .OnceSingleAsync<AdvancedPoints>()).fixCount;
58
59         fixCount2++;
60
61         await firebaseClient
62             .Child("AdvancedPoints")
63             .Child(auth.GetUid())
64             .PutAsync(new AdvancedPoints()
65                 {
66                     username = username,
67                     points = points2,
68                     numberOfLogs = numberOfLogs2,
69                     fixCount = fixCount2,
70                 });
71     }
72     catch (FirebaseException)
73     {
74         username = (await firebaseClient
75             .Child("users")
76             .Child(auth.GetUid())
77             .OnceSingleAsync<Users>()).username;
78
79         points2 = AppConstants.tenPoints;
80         await firebaseClient
81             .Child("AdvancedPoints")
82             .Child(auth.GetUid())
83             .PutAsync(new AdvancedPoints() { username = username, points = points2,
84 numberOfLogs = 1, fixCount = 1 }); ;
85     }
86     catch (NullReferenceException)
87     {
88         username = (await firebaseClient
89             .Child("users")
90             .Child(auth.GetUid())
91             .OnceSingleAsync<Users>()).username;
92
93         points2 = AppConstants.tenPoints;
94         await firebaseClient
95             .Child("AdvancedPoints")
96             .Child(auth.GetUid())
97             .PutAsync(new AdvancedPoints() { username = username, points = points2,
98 numberOfLogs = 1, fixCount = 1 });
99     }
100 }
101 }
```

```
1  /*! \class The CommunityPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the CommunityPointsUpdate ViewModel Class. It updates the data for
7  * the Community Category of the application. The functions in this class
8  * work by reading in all the chosen data and updating the selected fields and then sending
9  * this data to back firebase.
10 *
11 */
12 using Application_Green_Quake.Models;
13 using Firebase.Database;
14 using Firebase.Database.Query;
15 using System;
16 using Xamarin.Forms;
17
18 namespace Application_Green_Quake.ViewModels
19 {
20     class CommunityPointsUpdate
21     {
22         int points2 = 0;
23         int createGroupCount2 = 0;
24         int communityCount2 = 0;
25         int donateCount2 = 0;
26         int groupCount2 = 0;
27         int shareCount2 = 0;
28         int awarenessCount2 = 0;
29         int numberOfLogs2 = 0;
30
31         string username = "";
32
33         IAuth auth;
34         /** This function updates the points in the Community category by ten points. It
35         also increments the number of logs logged in the Community
36         * category by one and increments the number of times this particular action was
37         logged by one and sends this data to Firebase.
38         */
39         public async void CreateGroupPoints()
40         {
41             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
42             quake-default-rtdb.firebaseio.com/");
43             auth = DependencyService.Get<IAuth>();
44
45             try
46             {
47                 username = (await firebaseClient
48                 .Child("users")
49                 .Child(auth.GetUid())
50                 .OnceSingleAsync<Users>()).username;
51
52                 points2 = (await firebaseClient
53                 .Child("CommunityPoints")
54                 .Child(auth.GetUid())
55                 .OnceSingleAsync<CommunityPoints>()).points;
56
57                 points2 = points2 + AppConstants.tenPoints;
58
59                 numberOfLogs2 = (await firebaseClient
```

```
57     .Child("CommunityPoints")
58     .Child(auth.GetUid())
59     .OnceSingleAsync<CommunityPoints>().numberOfLogs;
60
61     numberOfLogs2++;
62
63     createGroupCount2 = (await firebaseClient
64     .Child("CommunityPoints")
65     .Child(auth.GetUid())
66     .OnceSingleAsync<CommunityPoints>()).createGroupCount;
67
68     createGroupCount2++;
69
70     communityCount2 = (await firebaseClient
71     .Child("CommunityPoints")
72     .Child(auth.GetUid())
73     .OnceSingleAsync<CommunityPoints>()).communityCount;
74
75     donateCount2 = (await firebaseClient
76     .Child("CommunityPoints")
77     .Child(auth.GetUid())
78     .OnceSingleAsync<CommunityPoints>()).donateCount;
79
80     groupCount2 = (await firebaseClient
81     .Child("CommunityPoints")
82     .Child(auth.GetUid())
83     .OnceSingleAsync<CommunityPoints>()).groupCount;
84
85     shareCount2 = (await firebaseClient
86     .Child("CommunityPoints")
87     .Child(auth.GetUid())
88     .OnceSingleAsync<CommunityPoints>()).shareCount;
89
90     awarenessCount2 = (await firebaseClient
91     .Child("CommunityPoints")
92     .Child(auth.GetUid())
93     .OnceSingleAsync<CommunityPoints>()).awarenessCount;
94
95     await firebaseClient
96     .Child("CommunityPoints")
97     .Child(auth.GetUid())
98     .PutAsync(new CommunityPoints()
99     {
100         username = username,
101         points = points2,
102         numberOfLogs = numberOfLogs2,
103         communityCount = communityCount2,
104         createGroupCount = createGroupCount2,
105         donateCount = donateCount2,
106         groupCount = groupCount2,
107         shareCount = shareCount2,
108         awarenessCount = awarenessCount2,
109
110     });
111 }
112 catch (FirebaseException)
113 {
114     username = (await firebaseClient
115     .Child("users")
116     .Child(auth.GetUid())
117     .OnceSingleAsync<Users>()).username;
```

```
118
119         points2 = AppConstants.tenPoints;
120         await firebaseClient
121             .Child("CommunityPoints")
122             .Child(auth.GetUid())
123             .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, createGroupCount = 1 }); ;
124     }
125     catch (NullReferenceException)
126     {
127         username = (await firebaseClient
128             .Child("users")
129             .Child(auth.GetUid())
130             .OnceSingleAsync<Users>()).username;
131
132         points2 = AppConstants.tenPoints;
133         await firebaseClient
134             .Child("CommunityPoints")
135             .Child(auth.GetUid())
136             .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, createGroupCount = 1 });
137     }
138 }
139
140 /** This function updates the points in the Community category by ten points. It
also increments the number of logs logged in the Community
141 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
142 */
143 public async void CommunityPoints()
144 {
145
146     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
147     auth = DependencyService.Get<IAuth>();
148
149     try
150     {
151         username = (await firebaseClient
152             .Child("users")
153             .Child(auth.GetUid())
154             .OnceSingleAsync<Users>()).username;
155
156         points2 = (await firebaseClient
157             .Child("CommunityPoints")
158             .Child(auth.GetUid())
159             .OnceSingleAsync<CommunityPoints>()).points;
160
161         points2 = points2 + AppConstants.tenPoints;
162
163         numberOfLogs2 = (await firebaseClient
164             .Child("CommunityPoints")
165             .Child(auth.GetUid())
166             .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
167
168         numberOfLogs2++;
169
170         createGroupCount2 = (await firebaseClient
171             .Child("CommunityPoints")
172             .Child(auth.GetUid())
173             .OnceSingleAsync<CommunityPoints>()).createGroupCount;
174
```

```
175     communityCount2 = (await firebaseClient
176     .Child("CommunityPoints")
177     .Child(auth.GetUid())
178     .OnceSingleAsync<CommunityPoints>()).communityCount;
179
180     communityCount2++;
181
182     donateCount2 = (await firebaseClient
183     .Child("CommunityPoints")
184     .Child(auth.GetUid())
185     .OnceSingleAsync<CommunityPoints>()).donateCount;
186
187     groupCount2 = (await firebaseClient
188     .Child("CommunityPoints")
189     .Child(auth.GetUid())
190     .OnceSingleAsync<CommunityPoints>()).groupCount;
191
192     shareCount2 = (await firebaseClient
193     .Child("CommunityPoints")
194     .Child(auth.GetUid())
195     .OnceSingleAsync<CommunityPoints>()).shareCount;
196
197     awarenessCount2 = (await firebaseClient
198     .Child("CommunityPoints")
199     .Child(auth.GetUid())
200     .OnceSingleAsync<CommunityPoints>()).awarenessCount;
201
202     await firebaseClient
203     .Child("CommunityPoints")
204     .Child(auth.GetUid())
205     .PutAsync(new CommunityPoints()
206     {
207         username = username,
208         points = points2,
209         numberOfLogs = numberOfLogs2,
210         communityCount = communityCount2,
211         createGroupCount = createGroupCount2,
212         donateCount = donateCount2,
213         groupCount = groupCount2,
214         shareCount = shareCount2,
215         awarenessCount = awarenessCount2,
216
217     });
218 }
219 catch (FirebaseException)
220 {
221     username = (await firebaseClient
222     .Child("users")
223     .Child(auth.GetUid())
224     .OnceSingleAsync<Users>()).username;
225
226     points2 = AppConstants.tenPoints;
227     await firebaseClient
228     .Child("CommunityPoints")
229     .Child(auth.GetUid())
230     .PutAsync(new CommunityPoints() { username = username, points = points2,
231 numberOfLogs = 1, communityCount = 1 }); ;
232 }
233 catch (NullReferenceException)
234 {
```

```
235         username = (await firebaseClient
236             .Child("users")
237             .Child(auth.GetUid())
238             .OnceSingleAsync<Users>()).username;
239
240         points2 = AppConstants.tenPoints;
241         await firebaseClient
242             .Child("CommunityPoints")
243             .Child(auth.GetUid())
244             .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, communityCount = 1 });
245     }
246 }
247 /** This function updates the points in the Community category by ten points. It
also increments the number of logs logged in the Community
248 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
249 */
250 public async void DonatePoints()
251 {
252
253     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
254     auth = DependencyService.Get<IAuth>();
255
256     try
257     {
258         username = (await firebaseClient
259             .Child("users")
260             .Child(auth.GetUid())
261             .OnceSingleAsync<Users>()).username;
262
263         points2 = (await firebaseClient
264             .Child("CommunityPoints")
265             .Child(auth.GetUid())
266             .OnceSingleAsync<CommunityPoints>()).points;
267
268         points2 = points2 + AppConstants.tenPoints;
269
270         numberOfLogs2 = (await firebaseClient
271             .Child("CommunityPoints")
272             .Child(auth.GetUid())
273             .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
274
275         numberOfLogs2++;
276
277         createGroupCount2 = (await firebaseClient
278             .Child("CommunityPoints")
279             .Child(auth.GetUid())
280             .OnceSingleAsync<CommunityPoints>()).createGroupCount;
281
282         communityCount2 = (await firebaseClient
283             .Child("CommunityPoints")
284             .Child(auth.GetUid())
285             .OnceSingleAsync<CommunityPoints>()).communityCount;
286
287         donateCount2 = (await firebaseClient
288             .Child("CommunityPoints")
289             .Child(auth.GetUid())
290             .OnceSingleAsync<CommunityPoints>()).donateCount;
291
292         donateCount2++;
```

```
293
294     groupCount2 = (await firebaseClient
295     .Child("CommunityPoints")
296     .Child(auth.GetUid())
297     .OnceSingleAsync<CommunityPoints>()).groupCount;
298
299     shareCount2 = (await firebaseClient
300     .Child("CommunityPoints")
301     .Child(auth.GetUid())
302     .OnceSingleAsync<CommunityPoints>()).shareCount;
303
304     awarenessCount2 = (await firebaseClient
305     .Child("CommunityPoints")
306     .Child(auth.GetUid())
307     .OnceSingleAsync<CommunityPoints>()).awarenessCount;
308
309     await firebaseClient
310     .Child("CommunityPoints")
311     .Child(auth.GetUid())
312     .PutAsync(new CommunityPoints()
313     {
314         username = username,
315         points = points2,
316         numberOfLogs = numberOfLogs2,
317         communityCount = communityCount2,
318         createGroupCount = createGroupCount2,
319         donateCount = donateCount2,
320         groupCount = groupCount2,
321         shareCount = shareCount2,
322         awarenessCount = awarenessCount2,
323
324     });
325 }
326 catch (FirebaseException)
327 {
328     username = (await firebaseClient
329     .Child("users")
330     .Child(auth.GetUid())
331     .OnceSingleAsync<Users>()).username;
332
333     points2 = AppConstants.tenPoints;
334     await firebaseClient
335     .Child("CommunityPoints")
336     .Child(auth.GetUid())
337     .PutAsync(new CommunityPoints() { username = username, points = points2,
338 numberOfLogs = 1, donateCount = 1 }); ;
339 }
340 catch (NullReferenceException)
341 {
342     username = (await firebaseClient
343     .Child("users")
344     .Child(auth.GetUid())
345     .OnceSingleAsync<Users>()).username;
346
347     points2 = AppConstants.tenPoints;
348     await firebaseClient
349     .Child("CommunityPoints")
350     .Child(auth.GetUid())
351     .PutAsync(new CommunityPoints() { username = username, points = points2,
352 numberOfLogs = 1, donateCount = 1 });
352 }
```

```
353     }
354     /** This function updates the points in the Community category by eight points. It
also increments the number of logs logged in the Community
355     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
356     */
357     public async void GroupPoints()
358     {
359
360         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
361         auth = DependencyService.Get<IAuth>();
362
363         try
364         {
365             username = (await firebaseClient
366                 .Child("users")
367                 .Child(auth.GetUid())
368                 .OnceSingleAsync<Users>()).username;
369
370             points2 = (await firebaseClient
371                 .Child("CommunityPoints")
372                 .Child(auth.GetUid())
373                 .OnceSingleAsync<CommunityPoints>()).points;
374
375             points2 = points2 + AppConstants.eightPoints;
376
377             numberOfLogs2 = (await firebaseClient
378                 .Child("CommunityPoints")
379                 .Child(auth.GetUid())
380                 .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
381
382             numberOfLogs2++;
383
384             createGroupCount2 = (await firebaseClient
385                 .Child("CommunityPoints")
386                 .Child(auth.GetUid())
387                 .OnceSingleAsync<CommunityPoints>()).createGroupCount;
388
389             communityCount2 = (await firebaseClient
390                 .Child("CommunityPoints")
391                 .Child(auth.GetUid())
392                 .OnceSingleAsync<CommunityPoints>()).communityCount;
393
394             donateCount2 = (await firebaseClient
395                 .Child("CommunityPoints")
396                 .Child(auth.GetUid())
397                 .OnceSingleAsync<CommunityPoints>()).donateCount;
398
399             groupCount2 = (await firebaseClient
400                 .Child("CommunityPoints")
401                 .Child(auth.GetUid())
402                 .OnceSingleAsync<CommunityPoints>()).groupCount;
403
404             groupCount2++;
405
406             shareCount2 = (await firebaseClient
407                 .Child("CommunityPoints")
408                 .Child(auth.GetUid())
409                 .OnceSingleAsync<CommunityPoints>()).shareCount;
410
411             awarenessCount2 = (await firebaseClient
```



```
412         .Child("CommunityPoints")
413         .Child(auth.GetUid())
414         .OnceSingleAsync<CommunityPoints>()).awarenessCount;
415
416         await firebaseClient
417         .Child("CommunityPoints")
418         .Child(auth.GetUid())
419         .PutAsync(new CommunityPoints()
420         {
421             username = username,
422             points = points2,
423             numberOfLogs = numberOfLogs2,
424             communityCount = communityCount2,
425             createGroupCount = createGroupCount2,
426             donateCount = donateCount2,
427             groupCount = groupCount2,
428             shareCount = shareCount2,
429             awarenessCount = awarenessCount2,
430
431         });
432     }
433     catch (FirebaseException)
434     {
435         username = (await firebaseClient
436         .Child("users")
437         .Child(auth.GetUid())
438         .OnceSingleAsync<Users>()).username;
439
440         points2 = AppConstants.eightPoints;
441         await firebaseClient
442         .Child("CommunityPoints")
443         .Child(auth.GetUid())
444         .PutAsync(new CommunityPoints() { username = username, points = points2,
445 numberOfLogs = 1, groupCount = 1 }); ;
446     }
447     catch (NullReferenceException)
448     {
449         username = (await firebaseClient
450         .Child("users")
451         .Child(auth.GetUid())
452         .OnceSingleAsync<Users>()).username;
453
454         points2 = AppConstants.eightPoints;
455         await firebaseClient
456         .Child("CommunityPoints")
457         .Child(auth.GetUid())
458         .PutAsync(new CommunityPoints() { username = username, points = points2,
459 numberOfLogs = 1, groupCount = 1 });
460     }
461     /** This function updates the points in the Community category by eight points. It
462     also increments the number of logs logged in the Community
463     * category by one and increments the number of times this particular action was
464     logged by one and sends this data to Firebase.
465     */
466     public async void SharePoints()
467     {
468         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
469 quake-default-rtdb.firebaseio.com/");
470         auth = DependencyService.Get<IAuth>();
```

```
469
470     try
471     {
472         username = (await firebaseClient
473             .Child("users")
474             .Child(auth.GetUid())
475             .OnceSingleAsync<Users>()).username;
476
477         points2 = (await firebaseClient
478             .Child("CommunityPoints")
479             .Child(auth.GetUid())
480             .OnceSingleAsync<CommunityPoints>()).points;
481
482         points2 = points2 + AppConstants.eightPoints;
483
484         numberOfLogs2 = (await firebaseClient
485             .Child("CommunityPoints")
486             .Child(auth.GetUid())
487             .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
488
489         numberOfLogs2++;
490
491         createGroupCount2 = (await firebaseClient
492             .Child("CommunityPoints")
493             .Child(auth.GetUid())
494             .OnceSingleAsync<CommunityPoints>()).createGroupCount;
495
496         communityCount2 = (await firebaseClient
497             .Child("CommunityPoints")
498             .Child(auth.GetUid())
499             .OnceSingleAsync<CommunityPoints>()).communityCount;
500
501         donateCount2 = (await firebaseClient
502             .Child("CommunityPoints")
503             .Child(auth.GetUid())
504             .OnceSingleAsync<CommunityPoints>()).donateCount;
505
506         groupCount2 = (await firebaseClient
507             .Child("CommunityPoints")
508             .Child(auth.GetUid())
509             .OnceSingleAsync<CommunityPoints>()).groupCount;
510
511         shareCount2 = (await firebaseClient
512             .Child("CommunityPoints")
513             .Child(auth.GetUid())
514             .OnceSingleAsync<CommunityPoints>()).shareCount;
515
516         shareCount2++;
517
518         awarenessCount2 = (await firebaseClient
519             .Child("CommunityPoints")
520             .Child(auth.GetUid())
521             .OnceSingleAsync<CommunityPoints>()).awarenessCount;
522
523         await firebaseClient
524             .Child("CommunityPoints")
525             .Child(auth.GetUid())
526             .PutAsync(new CommunityPoints()
527             {
528                 username = username,
529                 points = points2,
```

```
530         numberOfLogs = numberOfLogs2,
531         communityCount = communityCount2,
532         createGroupCount = createGroupCount2,
533         donateCount = donateCount2,
534         groupCount = groupCount2,
535         shareCount = shareCount2,
536         awarenessCount = awarenessCount2,
537     });
538 }
539 }
540 catch (FirebaseException)
541 {
542     username = (await firebaseClient
543     .Child("users")
544     .Child(auth.GetUid())
545     .OnceSingleAsync<Users>()).username;
546
547     points2 = AppConstants.eightPoints;
548     await firebaseClient
549     .Child("CommunityPoints")
550     .Child(auth.GetUid())
551     .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, shareCount = 1 }); ;
552 }
553 }
554 catch (NullReferenceException)
555 {
556     username = (await firebaseClient
557     .Child("users")
558     .Child(auth.GetUid())
559     .OnceSingleAsync<Users>()).username;
560
561     points2 = AppConstants.eightPoints;
562     await firebaseClient
563     .Child("CommunityPoints")
564     .Child(auth.GetUid())
565     .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, shareCount = 1 });
566 }
567 }
568 /** This function updates the points in the Community category by eight points. It
also increments the number of logs logged in the Community
569 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
570 */
571 public async void awarenessPoints()
572 {
573
574     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
575     auth = DependencyService.Get<IAuth>();
576
577     try
578     {
579         username = (await firebaseClient
580         .Child("users")
581         .Child(auth.GetUid())
582         .OnceSingleAsync<Users>()).username;
583
584         points2 = (await firebaseClient
585         .Child("CommunityPoints")
586         .Child(auth.GetUid())
```

```
587         .OnceSingleAsync<CommunityPoints>()).points;
588
589         points2 = points2 + AppConstants.eightPoints;
590
591         numberOfLogs2 = (await firebaseClient
592             .Child("CommunityPoints")
593             .Child(auth.GetUid())
594             .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
595
596         numberOfLogs2++;
597
598         createGroupCount2 = (await firebaseClient
599             .Child("CommunityPoints")
600             .Child(auth.GetUid())
601             .OnceSingleAsync<CommunityPoints>()).createGroupCount;
602
603         communityCount2 = (await firebaseClient
604             .Child("CommunityPoints")
605             .Child(auth.GetUid())
606             .OnceSingleAsync<CommunityPoints>()).communityCount;
607
608         donateCount2 = (await firebaseClient
609             .Child("CommunityPoints")
610             .Child(auth.GetUid())
611             .OnceSingleAsync<CommunityPoints>()).donateCount;
612
613         groupCount2 = (await firebaseClient
614             .Child("CommunityPoints")
615             .Child(auth.GetUid())
616             .OnceSingleAsync<CommunityPoints>()).groupCount;
617
618         shareCount2 = (await firebaseClient
619             .Child("CommunityPoints")
620             .Child(auth.GetUid())
621             .OnceSingleAsync<CommunityPoints>()).shareCount;
622
623         awarenessCount2 = (await firebaseClient
624             .Child("CommunityPoints")
625             .Child(auth.GetUid())
626             .OnceSingleAsync<CommunityPoints>()).awarenessCount;
627
628         awarenessCount2++;
629
630         await firebaseClient
631             .Child("CommunityPoints")
632             .Child(auth.GetUid())
633             .PutAsync(new CommunityPoints()
634             {
635                 username = username,
636                 points = points2,
637                 numberOfLogs = numberOfLogs2,
638                 communityCount = communityCount2,
639                 createGroupCount = createGroupCount2,
640                 donateCount = donateCount2,
641                 groupCount = groupCount2,
642                 shareCount = shareCount2,
643                 awarenessCount = awarenessCount2,
644
645             });
646     }
647     catch (FirebaseException)
```

```
648     {
649         username = (await firebaseClient
650             .Child("users")
651             .Child(auth.GetUid())
652             .OnceSingleAsync<Users>()).username;
653
654         points2 = AppConstants.eightPoints;
655         await firebaseClient
656             .Child("CommunityPoints")
657             .Child(auth.GetUid())
658             .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, awarenessCount = 1 }); ;
659     }
660 }
661 catch (NullReferenceException)
662 {
663     username = (await firebaseClient
664         .Child("users")
665         .Child(auth.GetUid())
666         .OnceSingleAsync<Users>()).username;
667
668     points2 = AppConstants.eightPoints;
669     await firebaseClient
670         .Child("CommunityPoints")
671         .Child(auth.GetUid())
672         .PutAsync(new CommunityPoints() { username = username, points = points2,
numberOfLogs = 1, awarenessCount = 1 }); ;
673     }
674 }
675 }
676 }
```

```

1  /*! \class The EnergyPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EnergyPointsUpdate ViewModel Class. It updates the data for
   the Energy Category of the application. The functions in this class
7  * work by reading in all the chosen data and updating the selected fields and then
   sending this data to back firebase.
8  *
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class EnergyPointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int hangDry2 = 0;
23         int fullDryerCount2 = 0;
24         int draftSealCount2 = 0;
25         int ductSealCount2 = 0;
26         int efficientThermostatCount2 = 0;
27         int fridgeCount2 = 0;
28         int fullMachineCount2 = 0;
29         int insulateWaterCount2 = 0;
30         int isolateHomeCount2 = 0;
31         int ledLightBulbCount2 = 0;
32         int microwaveCount2 = 0;
33         int solarPanelCount2 = 0;
34         int offSocketCount2 = 0;
35         int reBatteriesCount2 = 0;
36
37
38         string username = "";
39
40         IAuth auth;
41         /** This function updates the points in the Energy category by ten points. It also
   increments the number of logs logged in the Energy
42         * category by one and increments the number of times this particular action was
   logged by one and sends this data to Firebase.
43         */
44         public async void HangDryPoints()
45         {
46
47             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtadb.firebaseio.com/");
48             auth = DependencyService.Get<IAuth>();
49
50             try
51             {
52                 username = (await firebaseClient
53                     .Child("users")
54                     .Child(auth.GetUid())
55                     .OnceSingleAsync<Users>()).username;
56

```

```
57     points2 = (await firebaseClient
58         .Child("EnergyPoints")
59         .Child(auth.GetUid())
60         .OnceSingleAsync<EnergyPoints>()).points;
61
62     points2 = points2 + AppConstants.tenPoints;
63
64     numberOfLogs2 = (await firebaseClient
65         .Child("EnergyPoints")
66         .Child(auth.GetUid())
67         .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
68
69     numberOfLogs2++;
70
71     hangDry2 = (await firebaseClient
72         .Child("EnergyPoints")
73         .Child(auth.GetUid())
74         .OnceSingleAsync<EnergyPoints>()).hangDryCount;
75
76     hangDry2++;
77
78     draftSealCount2 = (await firebaseClient
79         .Child("EnergyPoints")
80         .Child(auth.GetUid())
81         .OnceSingleAsync<EnergyPoints>()).draftSealCount;
82
83     ductSealCount2 = (await firebaseClient
84         .Child("EnergyPoints")
85         .Child(auth.GetUid())
86         .OnceSingleAsync<EnergyPoints>()).ductSealCount;
87
88     efficientThermostatCount2 = (await firebaseClient
89         .Child("EnergyPoints")
90         .Child(auth.GetUid())
91         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
92
93     fridgeCount2 = (await firebaseClient
94         .Child("EnergyPoints")
95         .Child(auth.GetUid())
96         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
97
98     fullDryerCount2 = (await firebaseClient
99         .Child("EnergyPoints")
100        .Child(auth.GetUid())
101        .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
102
103     fullMachineCount2 = (await firebaseClient
104        .Child("EnergyPoints")
105        .Child(auth.GetUid())
106        .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
107
108     insulateWaterCount2 = (await firebaseClient
109        .Child("EnergyPoints")
110        .Child(auth.GetUid())
111        .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
112
113     ledLightBulbCount2 = (await firebaseClient
114        .Child("EnergyPoints")
115        .Child(auth.GetUid())
116        .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
117
```

```
118     microwaveCount2 = (await firebaseClient
119     .Child("EnergyPoints")
120     .Child(auth.GetUid())
121     .OnceSingleAsync<EnergyPoints>()).microwaveCount;
122
123     offSocketCount2 = (await firebaseClient
124     .Child("EnergyPoints")
125     .Child(auth.GetUid())
126     .OnceSingleAsync<EnergyPoints>()).offSocketCount;
127
128     reBatteriesCount2 = (await firebaseClient
129     .Child("EnergyPoints")
130     .Child(auth.GetUid())
131     .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
132
133     solarPanelCount2 = (await firebaseClient
134     .Child("EnergyPoints")
135     .Child(auth.GetUid())
136     .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
137
138     isolateHomeCount2 = (await firebaseClient
139     .Child("EnergyPoints")
140     .Child(auth.GetUid())
141     .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
142
143     await firebaseClient
144     .Child("EnergyPoints")
145     .Child(auth.GetUid())
146     .PutAsync(new EnergyPoints()
147     {
148         username = username,
149         points = points2,
150         numberOfLogs = numberOfLogs2,
151         hangDryCount = hangDry2,
152         draftSealCount = draftSealCount2,
153         ductSealCount = ductSealCount2,
154         efficientThermostatCount = efficientThermostatCount2,
155         fridgeCount = fridgeCount2,
156         fullDryerCount = fullDryerCount2,
157         insulateWaterCount = insulateWaterCount2,
158         isolateHomeCount = isolateHomeCount2,
159         ledLightBulbCount = ledLightBulbCount2,
160         microwaveCount = microwaveCount2,
161         offSocketCount = offSocketCount2,
162         reBatteriesCount = reBatteriesCount2,
163         solarPanelCount = solarPanelCount2,
164         fullMachineCount = fullMachineCount2
165     });
166 }
167 catch (FirebaseException)
168 {
169     username = (await firebaseClient
170     .Child("users")
171     .Child(auth.GetUid())
172     .OnceSingleAsync<Users>()).username;
173
174     points2 = AppConstants.tenPoints;
175     await firebaseClient
176     .Child("EnergyPoints")
177     .Child(auth.GetUid())
178     .PutAsync(new EnergyPoints() { username = username, points = points2,
```



```

179     numberOfLogs = 1, hangDryCount = 1});};
180     }
181     catch (NullReferenceException)
182     {
183         username = (await firebaseClient
184             .Child("users")
185             .Child(auth.GetUid())
186             .OnceSingleAsync<Users>()).username;
187
188         points2 = AppConstants.tenPoints;
189         await firebaseClient
190             .Child("EnergyPoints")
191             .Child(auth.GetUid())
192             .PutAsync(new EnergyPoints() { username = username, points = points2,
193                 numberOfLogs = 1, hangDryCount = 1 });
194     }
195     /** This function updates the points in the Energy category by ten points. It also
196     increments the number of logs logged in the Energy
197     * category by one and increments the number of times this particular action was
198     logged by one and sends this data to Firebase.
199     */
200     public async void DryerFullPoints()
201     {
202         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
203 quake-default-rtdb.firebaseio.com/");
204         auth = DependencyService.Get<IAuth>();
205
206         try
207         {
208             username = (await firebaseClient
209                 .Child("users")
210                 .Child(auth.GetUid())
211                 .OnceSingleAsync<Users>()).username;
212
213             points2 = (await firebaseClient
214                 .Child("EnergyPoints")
215                 .Child(auth.GetUid())
216                 .OnceSingleAsync<EnergyPoints>()).points;
217
218             points2 = points2 + AppConstants.tenPoints;
219
220             numberOfLogs2 = (await firebaseClient
221                 .Child("EnergyPoints")
222                 .Child(auth.GetUid())
223                 .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
224
225             numberOfLogs2++;
226
227             hangDry2 = (await firebaseClient
228                 .Child("EnergyPoints")
229                 .Child(auth.GetUid())
230                 .OnceSingleAsync<EnergyPoints>()).hangDryCount;
231
232             draftSealCount2 = (await firebaseClient
233                 .Child("EnergyPoints")
234                 .Child(auth.GetUid())
235                 .OnceSingleAsync<EnergyPoints>()).draftSealCount;

```

```
236         .Child("EnergyPoints")
237         .Child(auth.GetUid())
238         .OnceSingleAsync<EnergyPoints>()).ductSealCount;
239
240     efficientThermostatCount2 = (await firebaseClient
241         .Child("EnergyPoints")
242         .Child(auth.GetUid())
243         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
244
245     fridgeCount2 = (await firebaseClient
246         .Child("EnergyPoints")
247         .Child(auth.GetUid())
248         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
249
250     fullDryerCount2 = (await firebaseClient
251         .Child("EnergyPoints")
252         .Child(auth.GetUid())
253         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
254
255     fullDryerCount2++;
256
257     fullMachineCount2 = (await firebaseClient
258         .Child("EnergyPoints")
259         .Child(auth.GetUid())
260         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
261
262     insulateWaterCount2 = (await firebaseClient
263         .Child("EnergyPoints")
264         .Child(auth.GetUid())
265         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
266
267     ledLightBulbCount2 = (await firebaseClient
268         .Child("EnergyPoints")
269         .Child(auth.GetUid())
270         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
271
272     microwaveCount2 = (await firebaseClient
273         .Child("EnergyPoints")
274         .Child(auth.GetUid())
275         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
276
277     offSocketCount2 = (await firebaseClient
278         .Child("EnergyPoints")
279         .Child(auth.GetUid())
280         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
281
282     reBatteriesCount2 = (await firebaseClient
283         .Child("EnergyPoints")
284         .Child(auth.GetUid())
285         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
286
287     solarPanelCount2 = (await firebaseClient
288         .Child("EnergyPoints")
289         .Child(auth.GetUid())
290         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
291
292     isolateHomeCount2 = (await firebaseClient
293         .Child("EnergyPoints")
294         .Child(auth.GetUid())
295         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
296
```

```

297         await firebaseClient
298             .Child("EnergyPoints")
299             .Child(auth.GetUid())
300             .PutAsync(new EnergyPoints()
301             {
302                 username = username,
303                 points = points2,
304                 numberOfLogs = numberOfLogs2,
305                 hangDryCount = hangDry2,
306                 draftSealCount = draftSealCount2,
307                 ductSealCount = ductSealCount2,
308                 efficientThermostatCount = efficientThermostatCount2,
309                 fridgeCount = fridgeCount2,
310                 fullDryerCount = fullDryerCount2,
311                 insulateWaterCount = insulateWaterCount2,
312                 isolateHomeCount = isolateHomeCount2,
313                 ledLightBulbCount = ledLightBulbCount2,
314                 microwaveCount = microwaveCount2,
315                 offSocketCount = offSocketCount2,
316                 reBatteriesCount = reBatteriesCount2,
317                 solarPanelCount = solarPanelCount2,
318                 fullMachineCount = fullMachineCount2
319             });
320     }
321     catch (FirebaseException)
322     {
323         username = (await firebaseClient
324             .Child("users")
325             .Child(auth.GetUid())
326             .OnceSingleAsync<Users>()).username;
327
328         points2 = AppConstants.tenPoints;
329         await firebaseClient
330             .Child("EnergyPoints")
331             .Child(auth.GetUid())
332             .PutAsync(new EnergyPoints() { username = username, points = points2,
333 numberOfLogs = 1, fullDryerCount = 1 }); ;
334     }
335     catch (NullReferenceException)
336     {
337         username = (await firebaseClient
338             .Child("users")
339             .Child(auth.GetUid())
340             .OnceSingleAsync<Users>()).username;
341
342         points2 = AppConstants.tenPoints;
343         await firebaseClient
344             .Child("EnergyPoints")
345             .Child(auth.GetUid())
346             .PutAsync(new EnergyPoints() { username = username, points = points2,
347 numberOfLogs = 1, fullDryerCount = 1 }); ;
348     }
349     /** This function updates the points in the Energy category by eight points. It
350 also increments the number of logs logged in the Energy
351 * category by one and increments the number of times this particular action was
352 logged by one and sends this data to Firebase.
353 */
354     public async void EfficientThermostatPoints()
355     {

```

```
355     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
356     auth = DependencyService.Get<IAuth>();
357
358     try
359     {
360         username = (await firebaseClient
361             .Child("users")
362             .Child(auth.GetUid())
363             .OnceSingleAsync<Users>()).username;
364
365         points2 = (await firebaseClient
366             .Child("EnergyPoints")
367             .Child(auth.GetUid())
368             .OnceSingleAsync<EnergyPoints>()).points;
369
370         points2 = points2 + AppConstants.eightPoints;
371
372         numberOfLogs2 = (await firebaseClient
373             .Child("EnergyPoints")
374             .Child(auth.GetUid())
375             .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
376
377         numberOfLogs2++;
378
379         hangDry2 = (await firebaseClient
380             .Child("EnergyPoints")
381             .Child(auth.GetUid())
382             .OnceSingleAsync<EnergyPoints>()).hangDryCount;
383
384         draftSealCount2 = (await firebaseClient
385             .Child("EnergyPoints")
386             .Child(auth.GetUid())
387             .OnceSingleAsync<EnergyPoints>()).draftSealCount;
388
389         ductSealCount2 = (await firebaseClient
390             .Child("EnergyPoints")
391             .Child(auth.GetUid())
392             .OnceSingleAsync<EnergyPoints>()).ductSealCount;
393
394         efficientThermostatCount2 = (await firebaseClient
395             .Child("EnergyPoints")
396             .Child(auth.GetUid())
397             .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
398
399         efficientThermostatCount2++;
400
401         fridgeCount2 = (await firebaseClient
402             .Child("EnergyPoints")
403             .Child(auth.GetUid())
404             .OnceSingleAsync<EnergyPoints>()).fridgeCount;
405
406         fullDryerCount2 = (await firebaseClient
407             .Child("EnergyPoints")
408             .Child(auth.GetUid())
409             .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
410
411         fullMachineCount2 = (await firebaseClient
412             .Child("EnergyPoints")
413             .Child(auth.GetUid())
414             .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
```

```
415
416     insulateWaterCount2 = (await firebaseClient
417         .Child("EnergyPoints")
418         .Child(auth.GetUid())
419         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
420
421     ledLightBulbCount2 = (await firebaseClient
422         .Child("EnergyPoints")
423         .Child(auth.GetUid())
424         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
425
426     microwaveCount2 = (await firebaseClient
427         .Child("EnergyPoints")
428         .Child(auth.GetUid())
429         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
430
431     offSocketCount2 = (await firebaseClient
432         .Child("EnergyPoints")
433         .Child(auth.GetUid())
434         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
435
436     reBatteriesCount2 = (await firebaseClient
437         .Child("EnergyPoints")
438         .Child(auth.GetUid())
439         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
440
441     solarPanelCount2 = (await firebaseClient
442         .Child("EnergyPoints")
443         .Child(auth.GetUid())
444         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
445
446     isolateHomeCount2 = (await firebaseClient
447         .Child("EnergyPoints")
448         .Child(auth.GetUid())
449         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
450
451     await firebaseClient
452         .Child("EnergyPoints")
453         .Child(auth.GetUid())
454         .PutAsync(new EnergyPoints()
455     {
456         username = username,
457         points = points2,
458         numberOfLogs = numberOfLogs2,
459         hangDryCount = hangDry2,
460         draftSealCount = draftSealCount2,
461         ductSealCount = ductSealCount2,
462         efficientThermostatCount = efficientThermostatCount2,
463         fridgeCount = fridgeCount2,
464         fullDryerCount = fullDryerCount2,
465         insulateWaterCount = insulateWaterCount2,
466         isolateHomeCount = isolateHomeCount2,
467         ledLightBulbCount = ledLightBulbCount2,
468         microwaveCount = microwaveCount2,
469         offSocketCount = offSocketCount2,
470         reBatteriesCount = reBatteriesCount2,
471         solarPanelCount = solarPanelCount2,
472         fullMachineCount = fullMachineCount2
473     });
474 }
475 catch (FirebaseException)
```

```
476     {
477         username = (await firebaseClient
478             .Child("users")
479             .Child(auth.GetUid())
480             .OnceSingleAsync<Users>()).username;
481
482         points2 = AppConstants.eightPoints;
483         await firebaseClient
484             .Child("EnergyPoints")
485             .Child(auth.GetUid())
486             .PutAsync(new EnergyPoints() { username = username, points = points2,
487 numberOfLogs = 1, efficientThermostatCount = 1 }); ;
488     }
489     catch (NullReferenceException)
490     {
491         username = (await firebaseClient
492             .Child("users")
493             .Child(auth.GetUid())
494             .OnceSingleAsync<Users>()).username;
495
496         points2 = AppConstants.eightPoints;
497         await firebaseClient
498             .Child("EnergyPoints")
499             .Child(auth.GetUid())
500             .PutAsync(new EnergyPoints() { username = username, points = points2,
501 numberOfLogs = 1, efficientThermostatCount = 1 }); ;
502     }
503     /** This function updates the points in the Energy category by ten points. It also
504 increments the number of logs logged in the Energy
505 * category by one and increments the number of times this particular action was
506 logged by one and sends this data to Firebase.
507 */
508     public async void InsulateWaterPoints()
509     {
510         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
511 quake-default-rtdb.firebaseio.com/");
512         auth = DependencyService.Get<IAuth>();
513
514         try
515         {
516             username = (await firebaseClient
517                 .Child("users")
518                 .Child(auth.GetUid())
519                 .OnceSingleAsync<Users>()).username;
520
521             points2 = (await firebaseClient
522                 .Child("EnergyPoints")
523                 .Child(auth.GetUid())
524                 .OnceSingleAsync<EnergyPoints>()).points;
525
526             points2 = points2 + AppConstants.tenPoints;
527
528             numberOfLogs2 = (await firebaseClient
529                 .Child("EnergyPoints")
530                 .Child(auth.GetUid())
531                 .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
532
533             numberOfLogs2++;
534         }
535     }
536 }
```

```
533     hangDry2 = (await firebaseClient
534     .Child("EnergyPoints")
535     .Child(auth.GetUid())
536     .OnceSingleAsync<EnergyPoints>()).hangDryCount;
537
538     draftSealCount2 = (await firebaseClient
539     .Child("EnergyPoints")
540     .Child(auth.GetUid())
541     .OnceSingleAsync<EnergyPoints>()).draftSealCount;
542
543     ductSealCount2 = (await firebaseClient
544     .Child("EnergyPoints")
545     .Child(auth.GetUid())
546     .OnceSingleAsync<EnergyPoints>()).ductSealCount;
547
548     efficientThermostatCount2 = (await firebaseClient
549     .Child("EnergyPoints")
550     .Child(auth.GetUid())
551     .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
552
553     fridgeCount2 = (await firebaseClient
554     .Child("EnergyPoints")
555     .Child(auth.GetUid())
556     .OnceSingleAsync<EnergyPoints>()).fridgeCount;
557
558     fullDryerCount2 = (await firebaseClient
559     .Child("EnergyPoints")
560     .Child(auth.GetUid())
561     .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
562
563     fullMachineCount2 = (await firebaseClient
564     .Child("EnergyPoints")
565     .Child(auth.GetUid())
566     .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
567
568     insulateWaterCount2 = (await firebaseClient
569     .Child("EnergyPoints")
570     .Child(auth.GetUid())
571     .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
572
573     insulateWaterCount2++;
574
575     ledLightBulbCount2 = (await firebaseClient
576     .Child("EnergyPoints")
577     .Child(auth.GetUid())
578     .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
579
580     microwaveCount2 = (await firebaseClient
581     .Child("EnergyPoints")
582     .Child(auth.GetUid())
583     .OnceSingleAsync<EnergyPoints>()).microwaveCount;
584
585     offSocketCount2 = (await firebaseClient
586     .Child("EnergyPoints")
587     .Child(auth.GetUid())
588     .OnceSingleAsync<EnergyPoints>()).offSocketCount;
589
590     reBatteriesCount2 = (await firebaseClient
591     .Child("EnergyPoints")
592     .Child(auth.GetUid())
593     .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
```

```
594
595         solarPanelCount2 = (await firebaseClient
596             .Child("EnergyPoints")
597             .Child(auth.GetUid())
598             .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
599
600         isolateHomeCount2 = (await firebaseClient
601             .Child("EnergyPoints")
602             .Child(auth.GetUid())
603             .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
604
605         await firebaseClient
606             .Child("EnergyPoints")
607             .Child(auth.GetUid())
608             .PutAsync(new EnergyPoints()
609             {
610                 username = username,
611                 points = points2,
612                 numberOfLogs = numberOfLogs2,
613                 hangDryCount = hangDry2,
614                 draftSealCount = draftSealCount2,
615                 ductSealCount = ductSealCount2,
616                 efficientThermostatCount = efficientThermostatCount2,
617                 fridgeCount = fridgeCount2,
618                 fullDryerCount = fullDryerCount2,
619                 insulateWaterCount = insulateWaterCount2,
620                 isolateHomeCount = isolateHomeCount2,
621                 ledLightBulbCount = ledLightBulbCount2,
622                 microwaveCount = microwaveCount2,
623                 offSocketCount = offSocketCount2,
624                 reBatteriesCount = reBatteriesCount2,
625                 solarPanelCount = solarPanelCount2,
626                 fullMachineCount = fullMachineCount2
627             });
628     }
629     catch (FirebaseException)
630     {
631         username = (await firebaseClient
632             .Child("users")
633             .Child(auth.GetUid())
634             .OnceSingleAsync<Users>()).username;
635
636         points2 = AppConstants.tenPoints;
637         await firebaseClient
638             .Child("EnergyPoints")
639             .Child(auth.GetUid())
640             .PutAsync(new EnergyPoints() { username = username, points = points2,
641 numberOfLogs = 1, insulateWaterCount = 1 }); ;
642     }
643     catch (NullReferenceException)
644     {
645         username = (await firebaseClient
646             .Child("users")
647             .Child(auth.GetUid())
648             .OnceSingleAsync<Users>()).username;
649
650         points2 = AppConstants.tenPoints;
651         await firebaseClient
652             .Child("EnergyPoints")
653             .Child(auth.GetUid())
```



```
654         .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, insulateWaterCount = 1 });
655     }
656 }
657 /** This function updates the points in the Energy category by ten points. It also
increments the number of logs logged in the Energy
658     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
659     */
660     public async void IsolateHomePoints()
661     {
662
663         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
664         auth = DependencyService.Get<IAuth>();
665
666         try
667         {
668             username = (await firebaseClient
669                 .Child("users")
670                 .Child(auth.GetUid())
671                 .OnceSingleAsync<Users>()).username;
672
673             points2 = (await firebaseClient
674                 .Child("EnergyPoints")
675                 .Child(auth.GetUid())
676                 .OnceSingleAsync<EnergyPoints>()).points;
677
678             points2 = points2 + AppConstants.tenPoints;
679
680             numberOfLogs2 = (await firebaseClient
681                 .Child("EnergyPoints")
682                 .Child(auth.GetUid())
683                 .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
684
685             numberOfLogs2++;
686
687             hangDry2 = (await firebaseClient
688                 .Child("EnergyPoints")
689                 .Child(auth.GetUid())
690                 .OnceSingleAsync<EnergyPoints>()).hangDryCount;
691
692             draftSealCount2 = (await firebaseClient
693                 .Child("EnergyPoints")
694                 .Child(auth.GetUid())
695                 .OnceSingleAsync<EnergyPoints>()).draftSealCount;
696
697             ductSealCount2 = (await firebaseClient
698                 .Child("EnergyPoints")
699                 .Child(auth.GetUid())
700                 .OnceSingleAsync<EnergyPoints>()).ductSealCount;
701
702             efficientThermostatCount2 = (await firebaseClient
703                 .Child("EnergyPoints")
704                 .Child(auth.GetUid())
705                 .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
706
707             fridgeCount2 = (await firebaseClient
708                 .Child("EnergyPoints")
709                 .Child(auth.GetUid())
710                 .OnceSingleAsync<EnergyPoints>()).fridgeCount;
711
```

```
712         fullDryerCount2 = (await firebaseClient
713             .Child("EnergyPoints")
714             .Child(auth.GetUid())
715             .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
716
717         fullMachineCount2 = (await firebaseClient
718             .Child("EnergyPoints")
719             .Child(auth.GetUid())
720             .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
721
722         insulateWaterCount2 = (await firebaseClient
723             .Child("EnergyPoints")
724             .Child(auth.GetUid())
725             .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
726
727         ledLightBulbCount2 = (await firebaseClient
728             .Child("EnergyPoints")
729             .Child(auth.GetUid())
730             .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
731
732         microwaveCount2 = (await firebaseClient
733             .Child("EnergyPoints")
734             .Child(auth.GetUid())
735             .OnceSingleAsync<EnergyPoints>()).microwaveCount;
736
737         offSocketCount2 = (await firebaseClient
738             .Child("EnergyPoints")
739             .Child(auth.GetUid())
740             .OnceSingleAsync<EnergyPoints>()).offSocketCount;
741
742         reBatteriesCount2 = (await firebaseClient
743             .Child("EnergyPoints")
744             .Child(auth.GetUid())
745             .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
746
747         solarPanelCount2 = (await firebaseClient
748             .Child("EnergyPoints")
749             .Child(auth.GetUid())
750             .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
751
752         isolateHomeCount2 = (await firebaseClient
753             .Child("EnergyPoints")
754             .Child(auth.GetUid())
755             .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
756
757         isolateHomeCount2++;
758
759         await firebaseClient
760             .Child("EnergyPoints")
761             .Child(auth.GetUid())
762             .PutAsync(new EnergyPoints()
763             {
764                 username = username,
765                 points = points2,
766                 numberOfLogs = numberOfLogs2,
767                 hangDryCount = hangDry2,
768                 draftSealCount = draftSealCount2,
769                 ductSealCount = ductSealCount2,
770                 efficientThermostatCount = efficientThermostatCount2,
771                 fridgeCount = fridgeCount2,
772                 fullDryerCount = fullDryerCount2,
```

```

773         insulateWaterCount = insulateWaterCount2,
774         isolateHomeCount = isolateHomeCount2,
775         ledLightBulbCount = ledLightBulbCount2,
776         microwaveCount = microwaveCount2,
777         offSocketCount = offSocketCount2,
778         reBatteriesCount = reBatteriesCount2,
779         solarPanelCount = solarPanelCount2,
780         fullMachineCount = fullMachineCount2
781     });
782 }
783 catch (FirebaseException)
784 {
785     username = (await firebaseClient
786         .Child("users")
787         .Child(auth.GetUid())
788         .OnceSingleAsync<Users>()).username;
789
790     points2 = AppConstants.tenPoints;
791     await firebaseClient
792         .Child("EnergyPoints")
793         .Child(auth.GetUid())
794         .PutAsync(new EnergyPoints() { username = username, points = points2,
795 numberOfLogs = 1, isolateHomeCount = 1 }); ;
796 }
797 catch (NullReferenceException)
798 {
799     username = (await firebaseClient
800         .Child("users")
801         .Child(auth.GetUid())
802         .OnceSingleAsync<Users>()).username;
803
804     points2 = AppConstants.tenPoints;
805     await firebaseClient
806         .Child("EnergyPoints")
807         .Child(auth.GetUid())
808         .PutAsync(new EnergyPoints() { username = username, points = points2,
809 numberOfLogs = 1, isolateHomeCount = 1 }); ;
810 }
811 /** This function updates the points in the Energy category by ten points. It also
812 increments the number of logs logged in the Energy
813 * category by one and increments the number of times this particular action was
814 logged by one and sends this data to Firebase.
815 */
816 public async void LedLightsPoints()
817 {
818     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
819 quake-default-rtdb.firebaseio.com/");
820     auth = DependencyService.Get<IAuth>();
821
822     try
823     {
824         username = (await firebaseClient
825             .Child("users")
826             .Child(auth.GetUid())
827             .OnceSingleAsync<Users>()).username;
828
829         points2 = (await firebaseClient
830             .Child("EnergyPoints")
831             .Child(auth.GetUid())

```

```
830         .OnceSingleAsync<EnergyPoints>()).points;
831
832     points2 = points2 + AppConstants.tenPoints;
833
834     numberOfLogs2 = (await firebaseClient
835         .Child("EnergyPoints")
836         .Child(auth.GetUid())
837         .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
838
839     numberOfLogs2++;
840
841     hangDry2 = (await firebaseClient
842         .Child("EnergyPoints")
843         .Child(auth.GetUid())
844         .OnceSingleAsync<EnergyPoints>()).hangDryCount;
845
846     draftSealCount2 = (await firebaseClient
847         .Child("EnergyPoints")
848         .Child(auth.GetUid())
849         .OnceSingleAsync<EnergyPoints>()).draftSealCount;
850
851     ductSealCount2 = (await firebaseClient
852         .Child("EnergyPoints")
853         .Child(auth.GetUid())
854         .OnceSingleAsync<EnergyPoints>()).ductSealCount;
855
856     efficientThermostatCount2 = (await firebaseClient
857         .Child("EnergyPoints")
858         .Child(auth.GetUid())
859         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
860
861     fridgeCount2 = (await firebaseClient
862         .Child("EnergyPoints")
863         .Child(auth.GetUid())
864         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
865
866     fullDryerCount2 = (await firebaseClient
867         .Child("EnergyPoints")
868         .Child(auth.GetUid())
869         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
870
871     fullMachineCount2 = (await firebaseClient
872         .Child("EnergyPoints")
873         .Child(auth.GetUid())
874         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
875
876     insulateWaterCount2 = (await firebaseClient
877         .Child("EnergyPoints")
878         .Child(auth.GetUid())
879         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
880
881     ledLightBulbCount2 = (await firebaseClient
882         .Child("EnergyPoints")
883         .Child(auth.GetUid())
884         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
885
886     ledLightBulbCount2++;
887
888     microwaveCount2 = (await firebaseClient
889         .Child("EnergyPoints")
890         .Child(auth.GetUid())
```

```
891         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
892
893         offSocketCount2 = (await firebaseClient
894             .Child("EnergyPoints")
895             .Child(auth.GetUid())
896             .OnceSingleAsync<EnergyPoints>()).offSocketCount;
897
898         reBatteriesCount2 = (await firebaseClient
899             .Child("EnergyPoints")
900             .Child(auth.GetUid())
901             .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
902
903         solarPanelCount2 = (await firebaseClient
904             .Child("EnergyPoints")
905             .Child(auth.GetUid())
906             .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
907
908         isolateHomeCount2 = (await firebaseClient
909             .Child("EnergyPoints")
910             .Child(auth.GetUid())
911             .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
912
913         await firebaseClient
914             .Child("EnergyPoints")
915             .Child(auth.GetUid())
916             .PutAsync(new EnergyPoints()
917             {
918                 username = username,
919                 points = points2,
920                 numberOfLogs = numberOfLogs2,
921                 hangDryCount = hangDry2,
922                 draftSealCount = draftSealCount2,
923                 ductSealCount = ductSealCount2,
924                 efficientThermostatCount = efficientThermostatCount2,
925                 fridgeCount = fridgeCount2,
926                 fullDryerCount = fullDryerCount2,
927                 insulateWaterCount = insulateWaterCount2,
928                 isolateHomeCount = isolateHomeCount2,
929                 ledLightBulbCount = ledLightBulbCount2,
930                 microwaveCount = microwaveCount2,
931                 offSocketCount = offSocketCount2,
932                 reBatteriesCount = reBatteriesCount2,
933                 solarPanelCount = solarPanelCount2,
934                 fullMachineCount = fullMachineCount2
935             });
936     }
937     catch (FirebaseException)
938     {
939         username = (await firebaseClient
940             .Child("users")
941             .Child(auth.GetUid())
942             .OnceSingleAsync<Users>()).username;
943
944         points2 = AppConstants.tenPoints;
945         await firebaseClient
946             .Child("EnergyPoints")
947             .Child(auth.GetUid())
948             .PutAsync(new EnergyPoints() { username = username, points = points2,
949 numberOfLogs = 1, ledLightBulbCount = 1 }); ;
950     }
```

```
951     catch (NullReferenceException)
952     {
953         username = (await firebaseClient
954             .Child("users")
955             .Child(auth.GetUid())
956             .OnceSingleAsync<Users>()).username;
957
958         points2 = AppConstants.tenPoints;
959         await firebaseClient
960             .Child("EnergyPoints")
961             .Child(auth.GetUid())
962             .PutAsync(new EnergyPoints() { username = username, points = points2,
963 numberOfLogs = 1, ledLightBulbCount = 1 });
964     }
965     /** This function updates the points in the Energy category by eight points. It
966     also increments the number of logs logged in the Energy
967     * category by one and increments the number of times this particular action was
968     logged by one and sends this data to Firebase.
969     */
970     public async void MachineFullPoints()
971     {
972         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
973 quake-default-rtdb.firebaseio.com/");
974         auth = DependencyService.Get<IAuth>();
975
976         try
977         {
978             username = (await firebaseClient
979                 .Child("users")
980                 .Child(auth.GetUid())
981                 .OnceSingleAsync<Users>()).username;
982
983             points2 = (await firebaseClient
984                 .Child("EnergyPoints")
985                 .Child(auth.GetUid())
986                 .OnceSingleAsync<EnergyPoints>()).points;
987
988             points2 = points2 + AppConstants.eightPoints;
989
990             numberOfLogs2 = (await firebaseClient
991                 .Child("EnergyPoints")
992                 .Child(auth.GetUid())
993                 .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
994
995             numberOfLogs2++;
996
997             hangDry2 = (await firebaseClient
998                 .Child("EnergyPoints")
999                 .Child(auth.GetUid())
1000                 .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1001
1002             draftSealCount2 = (await firebaseClient
1003                 .Child("EnergyPoints")
1004                 .Child(auth.GetUid())
1005                 .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1006
1007             ductSealCount2 = (await firebaseClient
1008                 .Child("EnergyPoints")
1009                 .Child(auth.GetUid())
1010                 .OnceSingleAsync<EnergyPoints>()).ductSealCount;
```

```
1009
1010     efficientThermostatCount2 = (await firebaseClient
1011         .Child("EnergyPoints")
1012         .Child(auth.GetUid())
1013         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
1014
1015     fridgeCount2 = (await firebaseClient
1016         .Child("EnergyPoints")
1017         .Child(auth.GetUid())
1018         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1019
1020     fullDryerCount2 = (await firebaseClient
1021         .Child("EnergyPoints")
1022         .Child(auth.GetUid())
1023         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1024
1025     fullMachineCount2 = (await firebaseClient
1026         .Child("EnergyPoints")
1027         .Child(auth.GetUid())
1028         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
1029
1030     fullMachineCount2++;
1031
1032     insulateWaterCount2 = (await firebaseClient
1033         .Child("EnergyPoints")
1034         .Child(auth.GetUid())
1035         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
1036
1037     ledLightBulbCount2 = (await firebaseClient
1038         .Child("EnergyPoints")
1039         .Child(auth.GetUid())
1040         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
1041
1042     microwaveCount2 = (await firebaseClient
1043         .Child("EnergyPoints")
1044         .Child(auth.GetUid())
1045         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
1046
1047     offSocketCount2 = (await firebaseClient
1048         .Child("EnergyPoints")
1049         .Child(auth.GetUid())
1050         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
1051
1052     reBatteriesCount2 = (await firebaseClient
1053         .Child("EnergyPoints")
1054         .Child(auth.GetUid())
1055         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
1056
1057     solarPanelCount2 = (await firebaseClient
1058         .Child("EnergyPoints")
1059         .Child(auth.GetUid())
1060         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
1061
1062     isolateHomeCount2 = (await firebaseClient
1063         .Child("EnergyPoints")
1064         .Child(auth.GetUid())
1065         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1066
1067     await firebaseClient
1068         .Child("EnergyPoints")
1069         .Child(auth.GetUid())
```



```

1070         .PutAsync(new EnergyPoints()
1071         {
1072             username = username,
1073             points = points2,
1074             numberOfLogs = numberOfLogs2,
1075             hangDryCount = hangDry2,
1076             draftSealCount = draftSealCount2,
1077             ductSealCount = ductSealCount2,
1078             efficientThermostatCount = efficientThermostatCount2,
1079             fridgeCount = fridgeCount2,
1080             fullDryerCount = fullDryerCount2,
1081             insulateWaterCount = insulateWaterCount2,
1082             isolateHomeCount = isolateHomeCount2,
1083             ledLightBulbCount = ledLightBulbCount2,
1084             microwaveCount = microwaveCount2,
1085             offSocketCount = offSocketCount2,
1086             reBatteriesCount = reBatteriesCount2,
1087             solarPanelCount = solarPanelCount2,
1088             fullMachineCount = fullMachineCount2
1089         });
1090     }
1091     catch (FirebaseException)
1092     {
1093         username = (await firebaseClient
1094             .Child("users")
1095             .Child(auth.GetUid())
1096             .OnceSingleAsync<Users>()).username;
1097
1098         points2 = AppConstants.eightPoints;
1099         await firebaseClient
1100             .Child("EnergyPoints")
1101             .Child(auth.GetUid())
1102             .PutAsync(new EnergyPoints() { username = username, points = points2,
1103 numberOfLogs = 1, fullMachineCount = 1 }); ;
1104     }
1105     catch (NullReferenceException)
1106     {
1107         username = (await firebaseClient
1108             .Child("users")
1109             .Child(auth.GetUid())
1110             .OnceSingleAsync<Users>()).username;
1111
1112         points2 = AppConstants.eightPoints;
1113         await firebaseClient
1114             .Child("EnergyPoints")
1115             .Child(auth.GetUid())
1116             .PutAsync(new EnergyPoints() { username = username, points = points2,
1117 numberOfLogs = 1, fullMachineCount = 1 }); ;
1118     }
1119 }
1120 public async void MicrowavePoints()
1121 {
1122
1123     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1124 quake-default-rtdb.firebaseio.com/");
1125     auth = DependencyService.Get<IAuth>();
1126
1127     try
1128     {
1129         username = (await firebaseClient

```



```
1129         .Child("users")
1130         .Child(auth.GetUid())
1131         .OnceSingleAsync<Users>()).username;
1132
1133         points2 = (await firebaseClient
1134         .Child("EnergyPoints")
1135         .Child(auth.GetUid())
1136         .OnceSingleAsync<EnergyPoints>()).points;
1137
1138         points2 = points2 + AppConstants.fourPoints;
1139
1140         numberOfLogs2 = (await firebaseClient
1141         .Child("EnergyPoints")
1142         .Child(auth.GetUid())
1143         .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
1144
1145         numberOfLogs2++;
1146
1147         hangDry2 = (await firebaseClient
1148         .Child("EnergyPoints")
1149         .Child(auth.GetUid())
1150         .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1151
1152         draftSealCount2 = (await firebaseClient
1153         .Child("EnergyPoints")
1154         .Child(auth.GetUid())
1155         .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1156
1157         ductSealCount2 = (await firebaseClient
1158         .Child("EnergyPoints")
1159         .Child(auth.GetUid())
1160         .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1161
1162         efficientThermostatCount2 = (await firebaseClient
1163         .Child("EnergyPoints")
1164         .Child(auth.GetUid())
1165         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
1166
1167         fridgeCount2 = (await firebaseClient
1168         .Child("EnergyPoints")
1169         .Child(auth.GetUid())
1170         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1171
1172         fullDryerCount2 = (await firebaseClient
1173         .Child("EnergyPoints")
1174         .Child(auth.GetUid())
1175         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1176
1177         fullMachineCount2 = (await firebaseClient
1178         .Child("EnergyPoints")
1179         .Child(auth.GetUid())
1180         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
1181
1182         insulateWaterCount2 = (await firebaseClient
1183         .Child("EnergyPoints")
1184         .Child(auth.GetUid())
1185         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
1186
1187         ledLightBulbCount2 = (await firebaseClient
1188         .Child("EnergyPoints")
1189         .Child(auth.GetUid())
```

```
1190         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
1191
1192         microwaveCount2 = (await firebaseClient
1193         .Child("EnergyPoints")
1194         .Child(auth.GetUid())
1195         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
1196
1197         microwaveCount2++;
1198
1199         offSocketCount2 = (await firebaseClient
1200         .Child("EnergyPoints")
1201         .Child(auth.GetUid())
1202         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
1203
1204         reBatteriesCount2 = (await firebaseClient
1205         .Child("EnergyPoints")
1206         .Child(auth.GetUid())
1207         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
1208
1209         solarPanelCount2 = (await firebaseClient
1210         .Child("EnergyPoints")
1211         .Child(auth.GetUid())
1212         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
1213
1214         isolateHomeCount2 = (await firebaseClient
1215         .Child("EnergyPoints")
1216         .Child(auth.GetUid())
1217         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1218
1219         await firebaseClient
1220         .Child("EnergyPoints")
1221         .Child(auth.GetUid())
1222         .PutAsync(new EnergyPoints()
1223         {
1224             username = username,
1225             points = points2,
1226             numberOfLogs = numberOfLogs2,
1227             hangDryCount = hangDry2,
1228             draftSealCount = draftSealCount2,
1229             ductSealCount = ductSealCount2,
1230             efficientThermostatCount = efficientThermostatCount2,
1231             fridgeCount = fridgeCount2,
1232             fullDryerCount = fullDryerCount2,
1233             insulateWaterCount = insulateWaterCount2,
1234             isolateHomeCount = isolateHomeCount2,
1235             ledLightBulbCount = ledLightBulbCount2,
1236             microwaveCount = microwaveCount2,
1237             offSocketCount = offSocketCount2,
1238             reBatteriesCount = reBatteriesCount2,
1239             solarPanelCount = solarPanelCount2,
1240             fullMachineCount = fullMachineCount2
1241         });
1242     }
1243     catch (FirebaseException)
1244     {
1245         username = (await firebaseClient
1246         .Child("users")
1247         .Child(auth.GetUid())
1248         .OnceSingleAsync<Users>()).username;
1249
1250         points2 = AppConstants.fourPoints;
```

```
1251         await firebaseClient
1252             .Child("EnergyPoints")
1253             .Child(auth.GetUid())
1254             .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, microwaveCount = 1 }); ;
1255     }
1256 }
1257 catch (NullReferenceException)
1258 {
1259     username = (await firebaseClient
1260         .Child("users")
1261         .Child(auth.GetUid())
1262         .OnceSingleAsync<Users>()).username;
1263
1264     points2 = AppConstants.fourPoints;
1265     await firebaseClient
1266         .Child("EnergyPoints")
1267         .Child(auth.GetUid())
1268         .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, microwaveCount = 1 }); ;
1269     }
1270 }
1271 /** This function updates the points in the Energy category by four points. It
also increments the number of logs logged in the Energy
1272     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1273     */
1274     public async void SocketPoints()
1275     {
1276
1277         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1278         auth = DependencyService.Get<IAuth>();
1279
1280         try
1281         {
1282             username = (await firebaseClient
1283                 .Child("users")
1284                 .Child(auth.GetUid())
1285                 .OnceSingleAsync<Users>()).username;
1286
1287             points2 = (await firebaseClient
1288                 .Child("EnergyPoints")
1289                 .Child(auth.GetUid())
1290                 .OnceSingleAsync<EnergyPoints>()).points;
1291
1292             points2 = points2 + AppConstants.fourPoints;
1293
1294             numberOfLogs2 = (await firebaseClient
1295                 .Child("EnergyPoints")
1296                 .Child(auth.GetUid())
1297                 .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
1298
1299             numberOfLogs2++;
1300
1301             hangDry2 = (await firebaseClient
1302                 .Child("EnergyPoints")
1303                 .Child(auth.GetUid())
1304                 .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1305
1306             draftSealCount2 = (await firebaseClient
1307                 .Child("EnergyPoints")
```

```
1308         .Child(auth.GetUid())
1309         .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1310
1311         ductSealCount2 = (await firebaseClient
1312         .Child("EnergyPoints")
1313         .Child(auth.GetUid())
1314         .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1315
1316         efficientThermostatCount2 = (await firebaseClient
1317         .Child("EnergyPoints")
1318         .Child(auth.GetUid())
1319         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
1320
1321         fridgeCount2 = (await firebaseClient
1322         .Child("EnergyPoints")
1323         .Child(auth.GetUid())
1324         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1325
1326         fullDryerCount2 = (await firebaseClient
1327         .Child("EnergyPoints")
1328         .Child(auth.GetUid())
1329         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1330
1331         fullMachineCount2 = (await firebaseClient
1332         .Child("EnergyPoints")
1333         .Child(auth.GetUid())
1334         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
1335
1336         insulateWaterCount2 = (await firebaseClient
1337         .Child("EnergyPoints")
1338         .Child(auth.GetUid())
1339         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
1340
1341         ledLightBulbCount2 = (await firebaseClient
1342         .Child("EnergyPoints")
1343         .Child(auth.GetUid())
1344         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
1345
1346         microwaveCount2 = (await firebaseClient
1347         .Child("EnergyPoints")
1348         .Child(auth.GetUid())
1349         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
1350
1351         offSocketCount2 = (await firebaseClient
1352         .Child("EnergyPoints")
1353         .Child(auth.GetUid())
1354         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
1355
1356         offSocketCount2++;
1357
1358         reBatteriesCount2 = (await firebaseClient
1359         .Child("EnergyPoints")
1360         .Child(auth.GetUid())
1361         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
1362
1363         solarPanelCount2 = (await firebaseClient
1364         .Child("EnergyPoints")
1365         .Child(auth.GetUid())
1366         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
1367
1368         isolateHomeCount2 = (await firebaseClient
```

```
1369         .Child("EnergyPoints")
1370         .Child(auth.GetUid())
1371         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1372
1373         await firebaseClient
1374         .Child("EnergyPoints")
1375         .Child(auth.GetUid())
1376         .PutAsync(new EnergyPoints()
1377         {
1378             username = username,
1379             points = points2,
1380             numberOfLogs = numberOfLogs2,
1381             hangDryCount = hangDry2,
1382             draftSealCount = draftSealCount2,
1383             ductSealCount = ductSealCount2,
1384             efficientThermostatCount = efficientThermostatCount2,
1385             fridgeCount = fridgeCount2,
1386             fullDryerCount = fullDryerCount2,
1387             insulateWaterCount = insulateWaterCount2,
1388             isolateHomeCount = isolateHomeCount2,
1389             ledLightBulbCount = ledLightBulbCount2,
1390             microwaveCount = microwaveCount2,
1391             offSocketCount = offSocketCount2,
1392             reBatteriesCount = reBatteriesCount2,
1393             solarPanelCount = solarPanelCount2,
1394             fullMachineCount = fullMachineCount2
1395         });
1396     }
1397     catch (FirebaseException)
1398     {
1399         username = (await firebaseClient
1400         .Child("users")
1401         .Child(auth.GetUid())
1402         .OnceSingleAsync<Users>()).username;
1403
1404         points2 = AppConstants.fourPoints;
1405         await firebaseClient
1406         .Child("EnergyPoints")
1407         .Child(auth.GetUid())
1408         .PutAsync(new EnergyPoints() { username = username, points = points2,
1409 numberOfLogs = 1, offSocketCount = 1 }); ;
1410     }
1411     catch (NullReferenceException)
1412     {
1413         username = (await firebaseClient
1414         .Child("users")
1415         .Child(auth.GetUid())
1416         .OnceSingleAsync<Users>()).username;
1417
1418         points2 = AppConstants.fourPoints;
1419         await firebaseClient
1420         .Child("EnergyPoints")
1421         .Child(auth.GetUid())
1422         .PutAsync(new EnergyPoints() { username = username, points = points2,
1423 numberOfLogs = 1, offSocketCount = 1 }); ;
1424     }
1425     /** This function updates the points in the Energy category by six points. It also
1426     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.
```

```
1427 */
1428 public async void ReBatteriesPoints()
1429 {
1430
1431     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1432     auth = DependencyService.Get<IAuth>();
1433
1434     try
1435     {
1436         username = (await firebaseClient
1437             .Child("users")
1438             .Child(auth.GetUid())
1439             .OnceSingleAsync<Users>()).username;
1440
1441         points2 = (await firebaseClient
1442             .Child("EnergyPoints")
1443             .Child(auth.GetUid())
1444             .OnceSingleAsync<EnergyPoints>()).points;
1445
1446         points2 = points2 + AppConstants.sixPoints;
1447
1448         numberOfLogs2 = (await firebaseClient
1449             .Child("EnergyPoints")
1450             .Child(auth.GetUid())
1451             .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
1452
1453         numberOfLogs2++;
1454
1455         hangDry2 = (await firebaseClient
1456             .Child("EnergyPoints")
1457             .Child(auth.GetUid())
1458             .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1459
1460         draftSealCount2 = (await firebaseClient
1461             .Child("EnergyPoints")
1462             .Child(auth.GetUid())
1463             .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1464
1465         ductSealCount2 = (await firebaseClient
1466             .Child("EnergyPoints")
1467             .Child(auth.GetUid())
1468             .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1469
1470         efficientThermostatCount2 = (await firebaseClient
1471             .Child("EnergyPoints")
1472             .Child(auth.GetUid())
1473             .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
1474
1475         fridgeCount2 = (await firebaseClient
1476             .Child("EnergyPoints")
1477             .Child(auth.GetUid())
1478             .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1479
1480         fullDryerCount2 = (await firebaseClient
1481             .Child("EnergyPoints")
1482             .Child(auth.GetUid())
1483             .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1484
1485         fullMachineCount2 = (await firebaseClient
1486             .Child("EnergyPoints")
```

```
1487         .Child(auth.GetUid())
1488         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
1489
1490     insulateWaterCount2 = (await firebaseClient
1491         .Child("EnergyPoints")
1492         .Child(auth.GetUid())
1493         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
1494
1495     ledLightBulbCount2 = (await firebaseClient
1496         .Child("EnergyPoints")
1497         .Child(auth.GetUid())
1498         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
1499
1500     microwaveCount2 = (await firebaseClient
1501         .Child("EnergyPoints")
1502         .Child(auth.GetUid())
1503         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
1504
1505     offSocketCount2 = (await firebaseClient
1506         .Child("EnergyPoints")
1507         .Child(auth.GetUid())
1508         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
1509
1510     reBatteriesCount2 = (await firebaseClient
1511         .Child("EnergyPoints")
1512         .Child(auth.GetUid())
1513         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
1514
1515     reBatteriesCount2++;
1516
1517     solarPanelCount2 = (await firebaseClient
1518         .Child("EnergyPoints")
1519         .Child(auth.GetUid())
1520         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
1521
1522     isolateHomeCount2 = (await firebaseClient
1523         .Child("EnergyPoints")
1524         .Child(auth.GetUid())
1525         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1526
1527     await firebaseClient
1528         .Child("EnergyPoints")
1529         .Child(auth.GetUid())
1530         .PutAsync(new EnergyPoints()
1531     {
1532         username = username,
1533         points = points2,
1534         numberOfLogs = numberOfLogs2,
1535         hangDryCount = hangDry2,
1536         draftSealCount = draftSealCount2,
1537         ductSealCount = ductSealCount2,
1538         efficientThermostatCount = efficientThermostatCount2,
1539         fridgeCount = fridgeCount2,
1540         fullDryerCount = fullDryerCount2,
1541         insulateWaterCount = insulateWaterCount2,
1542         isolateHomeCount = isolateHomeCount2,
1543         ledLightBulbCount = ledLightBulbCount2,
1544         microwaveCount = microwaveCount2,
1545         offSocketCount = offSocketCount2,
1546         reBatteriesCount = reBatteriesCount2,
1547         solarPanelCount = solarPanelCount2,
```



```
1548         fullMachineCount = fullMachineCount2
1549     });
1550 }
1551 catch (FirebaseException)
1552 {
1553     username = (await firebaseClient
1554         .Child("users")
1555         .Child(auth.GetUid())
1556         .OnceSingleAsync<Users>()).username;
1557
1558     points2 = AppConstants.fourPoints;
1559     await firebaseClient
1560         .Child("EnergyPoints")
1561         .Child(auth.GetUid())
1562         .PutAsync(new EnergyPoints() { username = username, points = points2,
1563 numberOfLogs = 1, reBatteriesCount = 1 }); ;
1564 }
1565 catch (NullReferenceException)
1566 {
1567     username = (await firebaseClient
1568         .Child("users")
1569         .Child(auth.GetUid())
1570         .OnceSingleAsync<Users>()).username;
1571
1572     points2 = AppConstants.fourPoints;
1573     await firebaseClient
1574         .Child("EnergyPoints")
1575         .Child(auth.GetUid())
1576         .PutAsync(new EnergyPoints() { username = username, points = points2,
1577 numberOfLogs = 1, reBatteriesCount = 1 }); ;
1578 }
1579 /** This function updates the points in the Energy category by eight points. It
1580 also increments the number of logs logged in the Energy
1581 * category by one and increments the number of times this particular action was
1582 logged by one and sends this data to Firebase.
1583 */
1584 public async void FridgePoints()
1585 {
1586     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1587 quake-default-rtdb.firebaseio.com/");
1588     auth = DependencyService.Get<IAuth>();
1589
1590     try
1591     {
1592         username = (await firebaseClient
1593             .Child("users")
1594             .Child(auth.GetUid())
1595             .OnceSingleAsync<Users>()).username;
1596
1597         points2 = (await firebaseClient
1598             .Child("EnergyPoints")
1599             .Child(auth.GetUid())
1600             .OnceSingleAsync<EnergyPoints>()).points;
1601
1602         points2 = points2 + AppConstants.eightPoints;
1603
1604         numberOfLogs2 = (await firebaseClient
1605             .Child("EnergyPoints")
1606             .Child(auth.GetUid())
```



```
1605     .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
1606
1607     numberOfLogs2++;
1608
1609     hangDry2 = (await firebaseClient
1610     .Child("EnergyPoints")
1611     .Child(auth.GetUid())
1612     .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1613
1614     draftSealCount2 = (await firebaseClient
1615     .Child("EnergyPoints")
1616     .Child(auth.GetUid())
1617     .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1618
1619     ductSealCount2 = (await firebaseClient
1620     .Child("EnergyPoints")
1621     .Child(auth.GetUid())
1622     .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1623
1624     efficientThermostatCount2 = (await firebaseClient
1625     .Child("EnergyPoints")
1626     .Child(auth.GetUid())
1627     .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
1628
1629     fridgeCount2 = (await firebaseClient
1630     .Child("EnergyPoints")
1631     .Child(auth.GetUid())
1632     .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1633
1634     fridgeCount2++;
1635
1636     fullDryerCount2 = (await firebaseClient
1637     .Child("EnergyPoints")
1638     .Child(auth.GetUid())
1639     .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1640
1641     fullMachineCount2 = (await firebaseClient
1642     .Child("EnergyPoints")
1643     .Child(auth.GetUid())
1644     .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
1645
1646     insulateWaterCount2 = (await firebaseClient
1647     .Child("EnergyPoints")
1648     .Child(auth.GetUid())
1649     .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
1650
1651     ledLightBulbCount2 = (await firebaseClient
1652     .Child("EnergyPoints")
1653     .Child(auth.GetUid())
1654     .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
1655
1656     microwaveCount2 = (await firebaseClient
1657     .Child("EnergyPoints")
1658     .Child(auth.GetUid())
1659     .OnceSingleAsync<EnergyPoints>()).microwaveCount;
1660
1661     offSocketCount2 = (await firebaseClient
1662     .Child("EnergyPoints")
1663     .Child(auth.GetUid())
1664     .OnceSingleAsync<EnergyPoints>()).offSocketCount;
1665
```

```
1666         reBatteriesCount2 = (await firebaseClient
1667             .Child("EnergyPoints")
1668             .Child(auth.GetUid())
1669             .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
1670
1671         solarPanelCount2 = (await firebaseClient
1672             .Child("EnergyPoints")
1673             .Child(auth.GetUid())
1674             .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
1675
1676         isolateHomeCount2 = (await firebaseClient
1677             .Child("EnergyPoints")
1678             .Child(auth.GetUid())
1679             .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1680
1681         await firebaseClient
1682             .Child("EnergyPoints")
1683             .Child(auth.GetUid())
1684             .PutAsync(new EnergyPoints()
1685             {
1686                 username = username,
1687                 points = points2,
1688                 numberOfLogs = numberOfLogs2,
1689                 hangDryCount = hangDry2,
1690                 draftSealCount = draftSealCount2,
1691                 ductSealCount = ductSealCount2,
1692                 efficientThermostatCount = efficientThermostatCount2,
1693                 fridgeCount = fridgeCount2,
1694                 fullDryerCount = fullDryerCount2,
1695                 insulateWaterCount = insulateWaterCount2,
1696                 isolateHomeCount = isolateHomeCount2,
1697                 ledLightBulbCount = ledLightBulbCount2,
1698                 microwaveCount = microwaveCount2,
1699                 offSocketCount = offSocketCount2,
1700                 reBatteriesCount = reBatteriesCount2,
1701                 solarPanelCount = solarPanelCount2,
1702                 fullMachineCount = fullMachineCount2
1703             });
1704     }
1705     catch (FirebaseException)
1706     {
1707         username = (await firebaseClient
1708             .Child("users")
1709             .Child(auth.GetUid())
1710             .OnceSingleAsync<Users>()).username;
1711
1712         points2 = AppConstants.eightPoints;
1713         await firebaseClient
1714             .Child("EnergyPoints")
1715             .Child(auth.GetUid())
1716             .PutAsync(new EnergyPoints() { username = username, points = points2,
1717 numberOfLogs = 1, fridgeCount = 1 }); ;
1718     }
1719     catch (NullReferenceException)
1720     {
1721         username = (await firebaseClient
1722             .Child("users")
1723             .Child(auth.GetUid())
1724             .OnceSingleAsync<Users>()).username;
1725     }
```

```
1726         points2 = AppConstants.eightPoints;
1727         await firebaseClient
1728             .Child("EnergyPoints")
1729             .Child(auth.GetUid())
1730             .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, fridgeCount = 1 });
1731     }
1732 }
1733 /** This function updates the points in the Energy category by ten points. It also
increments the number of logs logged in the Energy
1734     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1735     */
1736     public async void SealDraftsPoints()
1737     {
1738
1739         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1740         auth = DependencyService.Get<IAuth>();
1741
1742         try
1743         {
1744             username = (await firebaseClient
1745                 .Child("users")
1746                 .Child(auth.GetUid())
1747                 .OnceSingleAsync<Users>()).username;
1748
1749             points2 = (await firebaseClient
1750                 .Child("EnergyPoints")
1751                 .Child(auth.GetUid())
1752                 .OnceSingleAsync<EnergyPoints>()).points;
1753
1754             points2 = points2 + AppConstants.tenPoints;
1755
1756             numberOfLogs2 = (await firebaseClient
1757                 .Child("EnergyPoints")
1758                 .Child(auth.GetUid())
1759                 .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
1760
1761             numberOfLogs2++;
1762
1763             hangDry2 = (await firebaseClient
1764                 .Child("EnergyPoints")
1765                 .Child(auth.GetUid())
1766                 .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1767
1768             draftSealCount2 = (await firebaseClient
1769                 .Child("EnergyPoints")
1770                 .Child(auth.GetUid())
1771                 .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1772
1773             draftSealCount2++;
1774
1775             ductSealCount2 = (await firebaseClient
1776                 .Child("EnergyPoints")
1777                 .Child(auth.GetUid())
1778                 .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1779
1780             efficientThermostatCount2 = (await firebaseClient
1781                 .Child("EnergyPoints")
1782                 .Child(auth.GetUid())
1783                 .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
```

```
1784
1785     fridgeCount2 = (await firebaseClient
1786         .Child("EnergyPoints")
1787         .Child(auth.GetUid())
1788         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1789
1790     fullDryerCount2 = (await firebaseClient
1791         .Child("EnergyPoints")
1792         .Child(auth.GetUid())
1793         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1794
1795     fullMachineCount2 = (await firebaseClient
1796         .Child("EnergyPoints")
1797         .Child(auth.GetUid())
1798         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
1799
1800     insulateWaterCount2 = (await firebaseClient
1801         .Child("EnergyPoints")
1802         .Child(auth.GetUid())
1803         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
1804
1805     ledLightBulbCount2 = (await firebaseClient
1806         .Child("EnergyPoints")
1807         .Child(auth.GetUid())
1808         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
1809
1810     microwaveCount2 = (await firebaseClient
1811         .Child("EnergyPoints")
1812         .Child(auth.GetUid())
1813         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
1814
1815     offSocketCount2 = (await firebaseClient
1816         .Child("EnergyPoints")
1817         .Child(auth.GetUid())
1818         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
1819
1820     reBatteriesCount2 = (await firebaseClient
1821         .Child("EnergyPoints")
1822         .Child(auth.GetUid())
1823         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
1824
1825     solarPanelCount2 = (await firebaseClient
1826         .Child("EnergyPoints")
1827         .Child(auth.GetUid())
1828         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
1829
1830     isolateHomeCount2 = (await firebaseClient
1831         .Child("EnergyPoints")
1832         .Child(auth.GetUid())
1833         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1834
1835     await firebaseClient
1836         .Child("EnergyPoints")
1837         .Child(auth.GetUid())
1838         .PutAsync(new EnergyPoints()
1839     {
1840         username = username,
1841         points = points2,
1842         numberOfLogs = numberOfLogs2,
1843         hangDryCount = hangDry2,
1844         draftSealCount = draftSealCount2,
```

```

1845         ductSealCount = ductSealCount2,
1846         efficientThermostatCount = efficientThermostatCount2,
1847         fridgeCount = fridgeCount2,
1848         fullDryerCount = fullDryerCount2,
1849         insulateWaterCount = insulateWaterCount2,
1850         isolateHomeCount = isolateHomeCount2,
1851         ledLightBulbCount = ledLightBulbCount2,
1852         microwaveCount = microwaveCount2,
1853         offSocketCount = offSocketCount2,
1854         reBatteriesCount = reBatteriesCount2,
1855         solarPanelCount = solarPanelCount2,
1856         fullMachineCount = fullMachineCount2
1857     });
1858 }
1859 catch (FirebaseException)
1860 {
1861     username = (await firebaseClient
1862         .Child("users")
1863         .Child(auth.GetUid())
1864         .OnceSingleAsync<Users>()).username;
1865
1866     points2 = AppConstants.tenPoints;
1867     await firebaseClient
1868         .Child("EnergyPoints")
1869         .Child(auth.GetUid())
1870         .PutAsync(new EnergyPoints() { username = username, points = points2,
1871 numberOfLogs = 1, draftSealCount = 1 }); ;
1872     }
1873 catch (NullReferenceException)
1874 {
1875     username = (await firebaseClient
1876         .Child("users")
1877         .Child(auth.GetUid())
1878         .OnceSingleAsync<Users>()).username;
1879
1880     points2 = AppConstants.tenPoints;
1881     await firebaseClient
1882         .Child("EnergyPoints")
1883         .Child(auth.GetUid())
1884         .PutAsync(new EnergyPoints() { username = username, points = points2,
1885 numberOfLogs = 1, draftSealCount = 1 });
1886     }
1887     /** This function updates the points in the Energy category by eight points. It
1888     * also increments the number of logs logged in the Energy
1889     * category by one and increments the number of times this particular action was
1890     * logged by one and sends this data to Firebase.
1891     */
1892     public async void SealDuctsPoints()
1893     {
1894         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
1895         auth = DependencyService.Get<IAuth>();
1896
1897         try
1898         {
1899             username = (await firebaseClient
1900                 .Child("users")
1901                 .Child(auth.GetUid())
1902                 .OnceSingleAsync<Users>()).username;

```

```
1902
1903     points2 = (await firebaseClient
1904         .Child("EnergyPoints")
1905         .Child(auth.GetUid())
1906         .OnceSingleAsync<EnergyPoints>()).points;
1907
1908     points2 = points2 + AppConstants.eightPoints;
1909
1910     numberOfLogs2 = (await firebaseClient
1911         .Child("EnergyPoints")
1912         .Child(auth.GetUid())
1913         .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
1914
1915     numberOfLogs2++;
1916
1917     hangDry2 = (await firebaseClient
1918         .Child("EnergyPoints")
1919         .Child(auth.GetUid())
1920         .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1921
1922     draftSealCount2 = (await firebaseClient
1923         .Child("EnergyPoints")
1924         .Child(auth.GetUid())
1925         .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1926
1927     ductSealCount2 = (await firebaseClient
1928         .Child("EnergyPoints")
1929         .Child(auth.GetUid())
1930         .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1931
1932     ductSealCount2++;
1933
1934     efficientThermostatCount2 = (await firebaseClient
1935         .Child("EnergyPoints")
1936         .Child(auth.GetUid())
1937         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
1938
1939     fridgeCount2 = (await firebaseClient
1940         .Child("EnergyPoints")
1941         .Child(auth.GetUid())
1942         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1943
1944     fullDryerCount2 = (await firebaseClient
1945         .Child("EnergyPoints")
1946         .Child(auth.GetUid())
1947         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1948
1949     fullMachineCount2 = (await firebaseClient
1950         .Child("EnergyPoints")
1951         .Child(auth.GetUid())
1952         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
1953
1954     insulateWaterCount2 = (await firebaseClient
1955         .Child("EnergyPoints")
1956         .Child(auth.GetUid())
1957         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
1958
1959     ledLightBulbCount2 = (await firebaseClient
1960         .Child("EnergyPoints")
1961         .Child(auth.GetUid())
1962         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
```

```
1963
1964     microwaveCount2 = (await firebaseClient
1965         .Child("EnergyPoints")
1966         .Child(auth.GetUid())
1967         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
1968
1969     offSocketCount2 = (await firebaseClient
1970         .Child("EnergyPoints")
1971         .Child(auth.GetUid())
1972         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
1973
1974     reBatteriesCount2 = (await firebaseClient
1975         .Child("EnergyPoints")
1976         .Child(auth.GetUid())
1977         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
1978
1979     solarPanelCount2 = (await firebaseClient
1980         .Child("EnergyPoints")
1981         .Child(auth.GetUid())
1982         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
1983
1984     isolateHomeCount2 = (await firebaseClient
1985         .Child("EnergyPoints")
1986         .Child(auth.GetUid())
1987         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1988
1989     await firebaseClient
1990         .Child("EnergyPoints")
1991         .Child(auth.GetUid())
1992         .PutAsync(new EnergyPoints()
1993         {
1994             username = username,
1995             points = points2,
1996             numberOfLogs = numberOfLogs2,
1997             hangDryCount = hangDry2,
1998             draftSealCount = draftSealCount2,
1999             ductSealCount = ductSealCount2,
2000             efficientThermostatCount = efficientThermostatCount2,
2001             fridgeCount = fridgeCount2,
2002             fullDryerCount = fullDryerCount2,
2003             insulateWaterCount = insulateWaterCount2,
2004             isolateHomeCount = isolateHomeCount2,
2005             ledLightBulbCount = ledLightBulbCount2,
2006             microwaveCount = microwaveCount2,
2007             offSocketCount = offSocketCount2,
2008             reBatteriesCount = reBatteriesCount2,
2009             solarPanelCount = solarPanelCount2,
2010             fullMachineCount = fullMachineCount2
2011         });
2012 }
2013 catch (FirebaseException)
2014 {
2015     username = (await firebaseClient
2016         .Child("users")
2017         .Child(auth.GetUid())
2018         .OnceSingleAsync<Users>()).username;
2019
2020     points2 = AppConstants.eightPoints;
2021     await firebaseClient
2022         .Child("EnergyPoints")
2023         .Child(auth.GetUid())
```



```
2024         .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, ductSealCount = 1 }); ;
2025
2026     }
2027     catch (NullReferenceException)
2028     {
2029         username = (await firebaseClient
2030             .Child("users")
2031             .Child(auth.GetUid())
2032             .OnceSingleAsync<Users>()).username;
2033
2034         points2 = AppConstants.eightPoints;
2035         await firebaseClient
2036             .Child("EnergyPoints")
2037             .Child(auth.GetUid())
2038             .PutAsync(new EnergyPoints() { username = username, points = points2,
numberOfLogs = 1, ductSealCount = 1 }); ;
2039     }
2040 }
2041 /** This function updates the points in the Energy category by ten points. It also
increments the number of logs logged in the Energy
2042 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
2043 */
2044 public async void SolarPanelPoints()
2045 {
2046
2047     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
2048     auth = DependencyService.Get<IAuth>();
2049
2050     try
2051     {
2052         username = (await firebaseClient
2053             .Child("users")
2054             .Child(auth.GetUid())
2055             .OnceSingleAsync<Users>()).username;
2056
2057         points2 = (await firebaseClient
2058             .Child("EnergyPoints")
2059             .Child(auth.GetUid())
2060             .OnceSingleAsync<EnergyPoints>()).points;
2061
2062         points2 = points2 + AppConstants.tenPoints;
2063
2064         numberOfLogs2 = (await firebaseClient
2065             .Child("EnergyPoints")
2066             .Child(auth.GetUid())
2067             .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
2068
2069         numberOfLogs2++;
2070
2071         hangDry2 = (await firebaseClient
2072             .Child("EnergyPoints")
2073             .Child(auth.GetUid())
2074             .OnceSingleAsync<EnergyPoints>()).hangDryCount;
2075
2076         draftSealCount2 = (await firebaseClient
2077             .Child("EnergyPoints")
2078             .Child(auth.GetUid())
2079             .OnceSingleAsync<EnergyPoints>()).draftSealCount;
2080
```



```
2081     ductSealCount2 = (await firebaseClient
2082     .Child("EnergyPoints")
2083     .Child(auth.GetUid())
2084     .OnceSingleAsync<EnergyPoints>()).ductSealCount;
2085
2086     efficientThermostatCount2 = (await firebaseClient
2087     .Child("EnergyPoints")
2088     .Child(auth.GetUid())
2089     .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
2090
2091     fridgeCount2 = (await firebaseClient
2092     .Child("EnergyPoints")
2093     .Child(auth.GetUid())
2094     .OnceSingleAsync<EnergyPoints>()).fridgeCount;
2095
2096     fullDryerCount2 = (await firebaseClient
2097     .Child("EnergyPoints")
2098     .Child(auth.GetUid())
2099     .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
2100
2101     fullMachineCount2 = (await firebaseClient
2102     .Child("EnergyPoints")
2103     .Child(auth.GetUid())
2104     .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
2105
2106     insulateWaterCount2 = (await firebaseClient
2107     .Child("EnergyPoints")
2108     .Child(auth.GetUid())
2109     .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
2110
2111     ledLightBulbCount2 = (await firebaseClient
2112     .Child("EnergyPoints")
2113     .Child(auth.GetUid())
2114     .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
2115
2116     microwaveCount2 = (await firebaseClient
2117     .Child("EnergyPoints")
2118     .Child(auth.GetUid())
2119     .OnceSingleAsync<EnergyPoints>()).microwaveCount;
2120
2121     offSocketCount2 = (await firebaseClient
2122     .Child("EnergyPoints")
2123     .Child(auth.GetUid())
2124     .OnceSingleAsync<EnergyPoints>()).offSocketCount;
2125
2126     reBatteriesCount2 = (await firebaseClient
2127     .Child("EnergyPoints")
2128     .Child(auth.GetUid())
2129     .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
2130
2131     solarPanelCount2 = (await firebaseClient
2132     .Child("EnergyPoints")
2133     .Child(auth.GetUid())
2134     .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
2135
2136     solarPanelCount2++;
2137
2138     isolateHomeCount2 = (await firebaseClient
2139     .Child("EnergyPoints")
2140     .Child(auth.GetUid())
2141     .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
```

```
2142
2143     await firebaseClient
2144     .Child("EnergyPoints")
2145     .Child(auth.GetUid())
2146     .PutAsync(new EnergyPoints()
2147     {
2148         username = username,
2149         points = points2,
2150         numberOfLogs = numberOfLogs2,
2151         hangDryCount = hangDry2,
2152         draftSealCount = draftSealCount2,
2153         ductSealCount = ductSealCount2,
2154         efficientThermostatCount = efficientThermostatCount2,
2155         fridgeCount = fridgeCount2,
2156         fullDryerCount = fullDryerCount2,
2157         insulateWaterCount = insulateWaterCount2,
2158         isolateHomeCount = isolateHomeCount2,
2159         ledLightBulbCount = ledLightBulbCount2,
2160         microwaveCount = microwaveCount2,
2161         offSocketCount = offSocketCount2,
2162         reBatteriesCount = reBatteriesCount2,
2163         solarPanelCount = solarPanelCount2,
2164         fullMachineCount = fullMachineCount2
2165     });
2166     }
2167     catch (FirebaseException)
2168     {
2169         username = (await firebaseClient
2170         .Child("users")
2171         .Child(auth.GetUid())
2172         .OnceSingleAsync<Users>()).username;
2173
2174         points2 = AppConstants.tenPoints;
2175         await firebaseClient
2176         .Child("EnergyPoints")
2177         .Child(auth.GetUid())
2178         .PutAsync(new EnergyPoints() { username = username, points = points2,
2179 numberOfLogs = 1, solarPanelCount = 1 }); ;
2180     }
2181     catch (NullReferenceException)
2182     {
2183         username = (await firebaseClient
2184         .Child("users")
2185         .Child(auth.GetUid())
2186         .OnceSingleAsync<Users>()).username;
2187
2188         points2 = AppConstants.tenPoints;
2189         await firebaseClient
2190         .Child("EnergyPoints")
2191         .Child(auth.GetUid())
2192         .PutAsync(new EnergyPoints() { username = username, points = points2,
2193 numberOfLogs = 1, solarPanelCount = 1 });
2194     }
2195     }
2196 }
```

```

1  /*! \class The FoodAndDrinkPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * 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
7  * work by reading in all the chosen data and updating the selected fields and then
   sending this data to back firebase.
8  *
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class FoodAndDrinkPointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int organicCount2 = 0;
23         int eatAllCount2 = 0;
24         int foodDeliverCount2 = 0;
25         int noMeatCount2 = 0;
26         int ownCoffeeCount2 = 0;
27         int reCoffeeMugCount2 = 0;
28         int saveLeftOversCount2 = 0;
29         int steelStrawCount2 = 0;
30         int waterOverFizzyCount2 = 0;
31
32         string username = "";
33
34         IAuth auth;
35         /** This function updates the points in the FoodAndDrink category by eight points.
   It also increments the number of logs logged in the FoodAndDrink
36         * category by one and increments the number of times this particular action was
   logged by one and sends this data to Firebase.
37         */
38         public async void OrganicPoints()
39         {
40
41             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtdb.firebaseio.com/");
42             auth = DependencyService.Get<IAuth>();
43
44             try
45             {
46                 username = (await firebaseClient
47                     .Child("users")
48                     .Child(auth.GetUid())
49                     .OnceSingleAsync<Users>()).username;
50
51                 points2 = (await firebaseClient
52                     .Child("FoodAndDrinkPoints")
53                     .Child(auth.GetUid())
54                     .OnceSingleAsync<FoodAndDrinkPoints>()).points;
55
56                 points2 = points2 + AppConstants.eightPoints;

```

```
57
58     numberOfLogs2 = (await firebaseClient
59         .Child("FoodAndDrinkPoints")
60         .Child(auth.GetUid())
61         .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
62
63     numberOfLogs2++;
64
65     organicCount2 = (await firebaseClient
66         .Child("FoodAndDrinkPoints")
67         .Child(auth.GetUid())
68         .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
69
70     organicCount2++;
71
72     eatAllCount2 = (await firebaseClient
73         .Child("FoodAndDrinkPoints")
74         .Child(auth.GetUid())
75         .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
76
77     foodDeliverCount2 = (await firebaseClient
78         .Child("FoodAndDrinkPoints")
79         .Child(auth.GetUid())
80         .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
81
82     noMeatCount2 = (await firebaseClient
83         .Child("FoodAndDrinkPoints")
84         .Child(auth.GetUid())
85         .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
86
87     ownCoffeeCount2 = (await firebaseClient
88         .Child("FoodAndDrinkPoints")
89         .Child(auth.GetUid())
90         .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
91
92     reCoffeeMugCount2 = (await firebaseClient
93         .Child("FoodAndDrinkPoints")
94         .Child(auth.GetUid())
95         .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
96
97     saveLeftOversCount2 = (await firebaseClient
98         .Child("FoodAndDrinkPoints")
99         .Child(auth.GetUid())
100        .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
101
102     steelStrawCount2 = (await firebaseClient
103         .Child("FoodAndDrinkPoints")
104         .Child(auth.GetUid())
105         .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
106
107     waterOverFizzyCount2 = (await firebaseClient
108         .Child("FoodAndDrinkPoints")
109         .Child(auth.GetUid())
110         .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
111
112     await firebaseClient
113         .Child("FoodAndDrinkPoints")
114         .Child(auth.GetUid())
115         .PutAsync(new FoodAndDrinkPoints()
116         {
117             username = username,
```

```

118         points = points2,
119         numberOfLogs = numberOfLogs2,
120         organicCount = organicCount2,
121         eatAllCount = eatAllCount2,
122         foodDeliverCount = foodDeliverCount2,
123         noMeatCount = noMeatCount2,
124         ownCoffeeCount = ownCoffeeCount2,
125         reCoffeeMugCount = reCoffeeMugCount2,
126         saveLeftOversCount = saveLeftOversCount2,
127         steelStrawCount = steelStrawCount2,
128         waterOverFizzyCount = waterOverFizzyCount2,
129     });
130 }
131 catch (FirebaseException)
132 {
133     username = (await firebaseClient
134         .Child("users")
135         .Child(auth.GetUid())
136         .OnceSingleAsync<Users>()).username;
137
138     points2 = AppConstants.eightPoints;
139     await firebaseClient
140         .Child("FoodAndDrinkPoints")
141         .Child(auth.GetUid())
142         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, organicCount = 1 }); ;
143
144     }
145     catch (NullReferenceException)
146     {
147         username = (await firebaseClient
148             .Child("users")
149             .Child(auth.GetUid())
150             .OnceSingleAsync<Users>()).username;
151
152         points2 = AppConstants.eightPoints;
153         await firebaseClient
154             .Child("FoodAndDrinkPoints")
155             .Child(auth.GetUid())
156             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, organicCount = 1 });
157     }
158 }
159 /** This function updates the points in the FoodAndDrink category by four points.
It also increments the number of logs logged in the FoodAndDrink
160 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
161 */
162 public async void EatAllPoints()
163 {
164
165     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
166     auth = DependencyService.Get<IAuth>();
167
168     try
169     {
170         username = (await firebaseClient
171             .Child("users")
172             .Child(auth.GetUid())
173             .OnceSingleAsync<Users>()).username;
174

```

```
175     points2 = (await firebaseClient
176     .Child("FoodAndDrinkPoints")
177     .Child(auth.GetUid())
178     .OnceSingleAsync<FoodAndDrinkPoints>()).points;
179
180     points2 = points2 + AppConstants.fourPoints;
181
182     numberOfLogs2 = (await firebaseClient
183     .Child("FoodAndDrinkPoints")
184     .Child(auth.GetUid())
185     .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
186
187     numberOfLogs2++;
188
189     organicCount2 = (await firebaseClient
190     .Child("FoodAndDrinkPoints")
191     .Child(auth.GetUid())
192     .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
193
194     eatAllCount2 = (await firebaseClient
195     .Child("FoodAndDrinkPoints")
196     .Child(auth.GetUid())
197     .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
198
199     eatAllCount2++;
200
201     foodDeliverCount2 = (await firebaseClient
202     .Child("FoodAndDrinkPoints")
203     .Child(auth.GetUid())
204     .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
205
206     noMeatCount2 = (await firebaseClient
207     .Child("FoodAndDrinkPoints")
208     .Child(auth.GetUid())
209     .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
210
211     ownCoffeeCount2 = (await firebaseClient
212     .Child("FoodAndDrinkPoints")
213     .Child(auth.GetUid())
214     .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
215
216     reCoffeeMugCount2 = (await firebaseClient
217     .Child("FoodAndDrinkPoints")
218     .Child(auth.GetUid())
219     .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
220
221     saveLeftOversCount2 = (await firebaseClient
222     .Child("FoodAndDrinkPoints")
223     .Child(auth.GetUid())
224     .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
225
226     steelStrawCount2 = (await firebaseClient
227     .Child("FoodAndDrinkPoints")
228     .Child(auth.GetUid())
229     .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
230
231     waterOverFizzyCount2 = (await firebaseClient
232     .Child("FoodAndDrinkPoints")
233     .Child(auth.GetUid())
234     .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
235
```

```

236         await firebaseClient
237         .Child("FoodAndDrinkPoints")
238         .Child(auth.GetUid())
239         .PutAsync(new FoodAndDrinkPoints()
240         {
241             username = username,
242             points = points2,
243             numberOfLogs = numberOfLogs2,
244             organicCount = organicCount2,
245             eatAllCount = eatAllCount2,
246             foodDeliverCount = foodDeliverCount2,
247             noMeatCount = noMeatCount2,
248             ownCoffeeCount = ownCoffeeCount2,
249             reCoffeeMugCount = reCoffeeMugCount2,
250             saveLeftOversCount = saveLeftOversCount2,
251             steelStrawCount = steelStrawCount2,
252             waterOverFizzyCount = waterOverFizzyCount2,
253         });
254     }
255     catch (FirebaseException)
256     {
257         username = (await firebaseClient
258         .Child("users")
259         .Child(auth.GetUid())
260         .OnceSingleAsync<Users>()).username;
261
262         points2 = AppConstants.fourPoints;
263         await firebaseClient
264         .Child("FoodAndDrinkPoints")
265         .Child(auth.GetUid())
266         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, eatAllCount = 1 }); ;
267     }
268     catch (NullReferenceException)
269     {
270
271         username = (await firebaseClient
272         .Child("users")
273         .Child(auth.GetUid())
274         .OnceSingleAsync<Users>()).username;
275
276         points2 = AppConstants.fourPoints;
277         await firebaseClient
278         .Child("FoodAndDrinkPoints")
279         .Child(auth.GetUid())
280         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, eatAllCount = 1 });;
281     }
282     }
283     /** This function updates the points in the FoodAndDrink category by six points.
It also increments the number of logs logged in the FoodAndDrink
284     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
285     */
286     public async void FoodDelivredPoints()
287     {
288
289         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
290         auth = DependencyService.Get<IAuth>();
291
292         try

```

```
293     {
294         username = (await firebaseClient
295             .Child("users")
296             .Child(auth.GetUid())
297             .OnceSingleAsync<Users>()).username;
298
299         points2 = (await firebaseClient
300             .Child("FoodAndDrinkPoints")
301             .Child(auth.GetUid())
302             .OnceSingleAsync<FoodAndDrinkPoints>()).points;
303
304         points2 = points2 + AppConstants.sixPoints;
305
306         numberOfLogs2 = (await firebaseClient
307             .Child("FoodAndDrinkPoints")
308             .Child(auth.GetUid())
309             .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
310
311         numberOfLogs2++;
312
313         organicCount2 = (await firebaseClient
314             .Child("FoodAndDrinkPoints")
315             .Child(auth.GetUid())
316             .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
317
318         eatAllCount2 = (await firebaseClient
319             .Child("FoodAndDrinkPoints")
320             .Child(auth.GetUid())
321             .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
322
323         foodDeliverCount2 = (await firebaseClient
324             .Child("FoodAndDrinkPoints")
325             .Child(auth.GetUid())
326             .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
327
328         foodDeliverCount2++;
329
330         noMeatCount2 = (await firebaseClient
331             .Child("FoodAndDrinkPoints")
332             .Child(auth.GetUid())
333             .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
334
335         ownCoffeeCount2 = (await firebaseClient
336             .Child("FoodAndDrinkPoints")
337             .Child(auth.GetUid())
338             .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
339
340         reCoffeeMugCount2 = (await firebaseClient
341             .Child("FoodAndDrinkPoints")
342             .Child(auth.GetUid())
343             .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
344
345         saveLeftOversCount2 = (await firebaseClient
346             .Child("FoodAndDrinkPoints")
347             .Child(auth.GetUid())
348             .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
349
350         steelStrawCount2 = (await firebaseClient
351             .Child("FoodAndDrinkPoints")
352             .Child(auth.GetUid())
353             .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
```



```

354
355         waterOverFizzyCount2 = (await firebaseClient
356             .Child("FoodAndDrinkPoints")
357             .Child(auth.GetUid())
358             .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
359
360         await firebaseClient
361             .Child("FoodAndDrinkPoints")
362             .Child(auth.GetUid())
363             .PutAsync(new FoodAndDrinkPoints()
364             {
365                 username = username,
366                 points = points2,
367                 numberOfLogs = numberOfLogs2,
368                 organicCount = organicCount2,
369                 eatAllCount = eatAllCount2,
370                 foodDeliverCount = foodDeliverCount2,
371                 noMeatCount = noMeatCount2,
372                 ownCoffeeCount = ownCoffeeCount2,
373                 reCoffeeMugCount = reCoffeeMugCount2,
374                 saveLeftOversCount = saveLeftOversCount2,
375                 steelStrawCount = steelStrawCount2,
376                 waterOverFizzyCount = waterOverFizzyCount2,
377             });
378     }
379     catch (FirebaseException)
380     {
381         username = (await firebaseClient
382             .Child("users")
383             .Child(auth.GetUid())
384             .OnceSingleAsync<Users>()).username;
385
386         points2 = AppConstants.sixPoints;
387         await firebaseClient
388             .Child("FoodAndDrinkPoints")
389             .Child(auth.GetUid())
390             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, foodDeliverCount = 1 }); ;
391     }
392     catch (NullReferenceException)
393     {
394         username = (await firebaseClient
395             .Child("users")
396             .Child(auth.GetUid())
397             .OnceSingleAsync<Users>()).username;
398
399         points2 = AppConstants.sixPoints;
400         await firebaseClient
401             .Child("FoodAndDrinkPoints")
402             .Child(auth.GetUid())
403             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, foodDeliverCount = 1 });
404     }
405     }
406     /** This function updates the points in the FoodAndDrink category by ten points.
It also increments the number of logs logged in the FoodAndDrink
408     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
409     */
410     public async void NoMeatPoints()
411     {

```

```
412
413     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
414     auth = DependencyService.Get<IAuth>();
415
416     try
417     {
418         username = (await firebaseClient
419             .Child("users")
420             .Child(auth.GetUid())
421             .OnceSingleAsync<Users>()).username;
422
423         points2 = (await firebaseClient
424             .Child("FoodAndDrinkPoints")
425             .Child(auth.GetUid())
426             .OnceSingleAsync<FoodAndDrinkPoints>()).points;
427
428         points2 = points2 + AppConstants.tenPoints;
429
430         numberOfLogs2 = (await firebaseClient
431             .Child("FoodAndDrinkPoints")
432             .Child(auth.GetUid())
433             .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
434
435         numberOfLogs2++;
436
437         organicCount2 = (await firebaseClient
438             .Child("FoodAndDrinkPoints")
439             .Child(auth.GetUid())
440             .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
441
442         eatAllCount2 = (await firebaseClient
443             .Child("FoodAndDrinkPoints")
444             .Child(auth.GetUid())
445             .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
446
447         foodDeliverCount2 = (await firebaseClient
448             .Child("FoodAndDrinkPoints")
449             .Child(auth.GetUid())
450             .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
451
452         noMeatCount2 = (await firebaseClient
453             .Child("FoodAndDrinkPoints")
454             .Child(auth.GetUid())
455             .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
456
457         noMeatCount2++;
458
459         ownCoffeeCount2 = (await firebaseClient
460             .Child("FoodAndDrinkPoints")
461             .Child(auth.GetUid())
462             .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
463
464         reCoffeeMugCount2 = (await firebaseClient
465             .Child("FoodAndDrinkPoints")
466             .Child(auth.GetUid())
467             .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
468
469         saveLeftOversCount2 = (await firebaseClient
470             .Child("FoodAndDrinkPoints")
471             .Child(auth.GetUid())
```

```

472         .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
473
474         steelStrawCount2 = (await firebaseClient
475             .Child("FoodAndDrinkPoints")
476             .Child(auth.GetUid())
477             .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
478
479         waterOverFizzyCount2 = (await firebaseClient
480             .Child("FoodAndDrinkPoints")
481             .Child(auth.GetUid())
482             .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
483
484         await firebaseClient
485             .Child("FoodAndDrinkPoints")
486             .Child(auth.GetUid())
487             .PutAsync(new FoodAndDrinkPoints()
488             {
489                 username = username,
490                 points = points2,
491                 numberOfLogs = numberOfLogs2,
492                 organicCount = organicCount2,
493                 eatAllCount = eatAllCount2,
494                 foodDeliverCount = foodDeliverCount2,
495                 noMeatCount = noMeatCount2,
496                 ownCoffeeCount = ownCoffeeCount2,
497                 reCoffeeMugCount = reCoffeeMugCount2,
498                 saveLeftOversCount = saveLeftOversCount2,
499                 steelStrawCount = steelStrawCount2,
500                 waterOverFizzyCount = waterOverFizzyCount2,
501             });
502     }
503     catch (FirebaseException)
504     {
505         username = (await firebaseClient
506             .Child("users")
507             .Child(auth.GetUid())
508             .OnceSingleAsync<Users>()).username;
509
510         points2 = AppConstants.tenPoints;
511         await firebaseClient
512             .Child("FoodAndDrinkPoints")
513             .Child(auth.GetUid())
514             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, noMeatCount = 1 }); ;
515     }
516     catch (NullReferenceException)
517     {
518         username = (await firebaseClient
519             .Child("users")
520             .Child(auth.GetUid())
521             .OnceSingleAsync<Users>()).username;
522
523         points2 = AppConstants.tenPoints;
524         await firebaseClient
525             .Child("FoodAndDrinkPoints")
526             .Child(auth.GetUid())
527             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, noMeatCount = 1 }); ;
528     }
529     }
530 }
531 /** This function updates the points in the FoodAndDrink category by two points.

```

```
It also increments the number of logs logged in the FoodAndDrink
532     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
533     */
534     public async void OwnCoffeePoints()
535     {
536
537         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
538         auth = DependencyService.Get<IAuth>();
539
540         try
541         {
542             username = (await firebaseClient
543                 .Child("users")
544                 .Child(auth.GetUid())
545                 .OnceSingleAsync<Users>()).username;
546
547             points2 = (await firebaseClient
548                 .Child("FoodAndDrinkPoints")
549                 .Child(auth.GetUid())
550                 .OnceSingleAsync<FoodAndDrinkPoints>()).points;
551
552             points2 = points2 + AppConstants.twoPoints;
553
554             numberOfLogs2 = (await firebaseClient
555                 .Child("FoodAndDrinkPoints")
556                 .Child(auth.GetUid())
557                 .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
558
559             numberOfLogs2++;
560
561             organicCount2 = (await firebaseClient
562                 .Child("FoodAndDrinkPoints")
563                 .Child(auth.GetUid())
564                 .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
565
566             eatAllCount2 = (await firebaseClient
567                 .Child("FoodAndDrinkPoints")
568                 .Child(auth.GetUid())
569                 .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
570
571             foodDeliverCount2 = (await firebaseClient
572                 .Child("FoodAndDrinkPoints")
573                 .Child(auth.GetUid())
574                 .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
575
576             noMeatCount2 = (await firebaseClient
577                 .Child("FoodAndDrinkPoints")
578                 .Child(auth.GetUid())
579                 .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
580
581             ownCoffeeCount2 = (await firebaseClient
582                 .Child("FoodAndDrinkPoints")
583                 .Child(auth.GetUid())
584                 .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
585
586             ownCoffeeCount2++;
587
588             reCoffeeMugCount2 = (await firebaseClient
589                 .Child("FoodAndDrinkPoints")
590                 .Child(auth.GetUid())
```

```
591         .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
592
593         saveLeftOversCount2 = (await firebaseClient
594             .Child("FoodAndDrinkPoints")
595             .Child(auth.GetUid())
596             .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
597
598         steelStrawCount2 = (await firebaseClient
599             .Child("FoodAndDrinkPoints")
600             .Child(auth.GetUid())
601             .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
602
603         waterOverFizzyCount2 = (await firebaseClient
604             .Child("FoodAndDrinkPoints")
605             .Child(auth.GetUid())
606             .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
607
608         await firebaseClient
609             .Child("FoodAndDrinkPoints")
610             .Child(auth.GetUid())
611             .PutAsync(new FoodAndDrinkPoints()
612             {
613                 username = username,
614                 points = points2,
615                 numberOfLogs = numberOfLogs2,
616                 organicCount = organicCount2,
617                 eatAllCount = eatAllCount2,
618                 foodDeliverCount = foodDeliverCount2,
619                 noMeatCount = noMeatCount2,
620                 ownCoffeeCount = ownCoffeeCount2,
621                 reCoffeeMugCount = reCoffeeMugCount2,
622                 saveLeftOversCount = saveLeftOversCount2,
623                 steelStrawCount = steelStrawCount2,
624                 waterOverFizzyCount = waterOverFizzyCount2,
625             });
626     }
627     catch (FirebaseException)
628     {
629         username = (await firebaseClient
630             .Child("users")
631             .Child(auth.GetUid())
632             .OnceSingleAsync<Users>()).username;
633
634         points2 = AppConstants.twoPoints;
635         await firebaseClient
636             .Child("FoodAndDrinkPoints")
637             .Child(auth.GetUid())
638             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, ownCoffeeCount = 1 }); ;
639     }
640     catch (NullReferenceException)
641     {
642         username = (await firebaseClient
643             .Child("users")
644             .Child(auth.GetUid())
645             .OnceSingleAsync<Users>()).username;
646
647         points2 = AppConstants.twoPoints;
648         await firebaseClient
649             .Child("FoodAndDrinkPoints")
```

```
651         .Child(auth.GetUid())
652         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, ownCoffeeCount = 1 });
653     }
654 }
655 /** This function updates the points in the FoodAndDrink category by four points.
It also increments the number of logs logged in the FoodAndDrink
656     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
657     */
658     public async void ReCoffeeMugPoints()
659     {
660
661         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
662         auth = DependencyService.Get<IAuth>();
663
664         try
665         {
666             username = (await firebaseClient
667                 .Child("users")
668                 .Child(auth.GetUid())
669                 .OnceSingleAsync<Users>()).username;
670
671             points2 = (await firebaseClient
672                 .Child("FoodAndDrinkPoints")
673                 .Child(auth.GetUid())
674                 .OnceSingleAsync<FoodAndDrinkPoints>()).points;
675
676             points2 = points2 + AppConstants.fourPoints;
677
678             numberOfLogs2 = (await firebaseClient
679                 .Child("FoodAndDrinkPoints")
680                 .Child(auth.GetUid())
681                 .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
682
683             numberOfLogs2++;
684
685             organicCount2 = (await firebaseClient
686                 .Child("FoodAndDrinkPoints")
687                 .Child(auth.GetUid())
688                 .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
689
690             eatAllCount2 = (await firebaseClient
691                 .Child("FoodAndDrinkPoints")
692                 .Child(auth.GetUid())
693                 .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
694
695             foodDeliverCount2 = (await firebaseClient
696                 .Child("FoodAndDrinkPoints")
697                 .Child(auth.GetUid())
698                 .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
699
700             noMeatCount2 = (await firebaseClient
701                 .Child("FoodAndDrinkPoints")
702                 .Child(auth.GetUid())
703                 .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
704
705             ownCoffeeCount2 = (await firebaseClient
706                 .Child("FoodAndDrinkPoints")
707                 .Child(auth.GetUid())
708                 .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
```

```

709
710         reCoffeeMugCount2 = (await firebaseClient
711             .Child("FoodAndDrinkPoints")
712             .Child(auth.GetUid())
713             .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
714
715         reCoffeeMugCount2++;
716
717         saveLeftOversCount2 = (await firebaseClient
718             .Child("FoodAndDrinkPoints")
719             .Child(auth.GetUid())
720             .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
721
722         steelStrawCount2 = (await firebaseClient
723             .Child("FoodAndDrinkPoints")
724             .Child(auth.GetUid())
725             .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
726
727         waterOverFizzyCount2 = (await firebaseClient
728             .Child("FoodAndDrinkPoints")
729             .Child(auth.GetUid())
730             .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
731
732         await firebaseClient
733             .Child("FoodAndDrinkPoints")
734             .Child(auth.GetUid())
735             .PutAsync(new FoodAndDrinkPoints()
736             {
737                 username = username,
738                 points = points2,
739                 numberOfLogs = numberOfLogs2,
740                 organicCount = organicCount2,
741                 eatAllCount = eatAllCount2,
742                 foodDeliverCount = foodDeliverCount2,
743                 noMeatCount = noMeatCount2,
744                 ownCoffeeCount = ownCoffeeCount2,
745                 reCoffeeMugCount = reCoffeeMugCount2,
746                 saveLeftOversCount = saveLeftOversCount2,
747                 steelStrawCount = steelStrawCount2,
748                 waterOverFizzyCount = waterOverFizzyCount2,
749             });
750     }
751     catch (FirebaseException)
752     {
753         username = (await firebaseClient
754             .Child("users")
755             .Child(auth.GetUid())
756             .OnceSingleAsync<Users>()).username;
757
758         points2 = AppConstants.fourPoints;
759         await firebaseClient
760             .Child("FoodAndDrinkPoints")
761             .Child(auth.GetUid())
762             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, reCoffeeMugCount = 1 }); ;
763     }
764     catch (NullReferenceException)
765     {
766         username = (await firebaseClient
767             .Child("users")

```



```
769         .Child(auth.GetUid())
770         .OnceSingleAsync<Users>()).username;
771
772         points2 = AppConstants.fourPoints;
773         await firebaseClient
774         .Child("FoodAndDrinkPoints")
775         .Child(auth.GetUid())
776         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, reCoffeeMugCount = 1 });
777     }
778 }
779 /** This function updates the points in the FoodAndDrink category by six points.
It also increments the number of logs logged in the FoodAndDrink
780 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
781 */
782 public async void SaveLeftOversPoints()
783 {
784
785     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
786     auth = DependencyService.Get<IAuth>();
787
788     try
789     {
790         username = (await firebaseClient
791         .Child("users")
792         .Child(auth.GetUid())
793         .OnceSingleAsync<Users>()).username;
794
795         points2 = (await firebaseClient
796         .Child("FoodAndDrinkPoints")
797         .Child(auth.GetUid())
798         .OnceSingleAsync<FoodAndDrinkPoints>()).points;
799
800         points2 = points2 + AppConstants.sixPoints;
801
802         numberOfLogs2 = (await firebaseClient
803         .Child("FoodAndDrinkPoints")
804         .Child(auth.GetUid())
805         .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
806
807         numberOfLogs2++;
808
809         organicCount2 = (await firebaseClient
810         .Child("FoodAndDrinkPoints")
811         .Child(auth.GetUid())
812         .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
813
814         eatAllCount2 = (await firebaseClient
815         .Child("FoodAndDrinkPoints")
816         .Child(auth.GetUid())
817         .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
818
819         foodDeliverCount2 = (await firebaseClient
820         .Child("FoodAndDrinkPoints")
821         .Child(auth.GetUid())
822         .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
823
824         noMeatCount2 = (await firebaseClient
825         .Child("FoodAndDrinkPoints")
826         .Child(auth.GetUid())
```



```
827         .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
828
829         ownCoffeeCount2 = (await firebaseClient
830         .Child("FoodAndDrinkPoints")
831         .Child(auth.GetUid())
832         .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
833
834         reCoffeeMugCount2 = (await firebaseClient
835         .Child("FoodAndDrinkPoints")
836         .Child(auth.GetUid())
837         .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
838
839         saveLeftOversCount2 = (await firebaseClient
840         .Child("FoodAndDrinkPoints")
841         .Child(auth.GetUid())
842         .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
843
844         saveLeftOversCount2++;
845
846         steelStrawCount2 = (await firebaseClient
847         .Child("FoodAndDrinkPoints")
848         .Child(auth.GetUid())
849         .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
850
851         waterOverFizzyCount2 = (await firebaseClient
852         .Child("FoodAndDrinkPoints")
853         .Child(auth.GetUid())
854         .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
855
856         await firebaseClient
857         .Child("FoodAndDrinkPoints")
858         .Child(auth.GetUid())
859         .PutAsync(new FoodAndDrinkPoints()
860         {
861             username = username,
862             points = points2,
863             numberOfLogs = numberOfLogs2,
864             organicCount = organicCount2,
865             eatAllCount = eatAllCount2,
866             foodDeliverCount = foodDeliverCount2,
867             noMeatCount = noMeatCount2,
868             ownCoffeeCount = ownCoffeeCount2,
869             reCoffeeMugCount = reCoffeeMugCount2,
870             saveLeftOversCount = saveLeftOversCount2,
871             steelStrawCount = steelStrawCount2,
872             waterOverFizzyCount = waterOverFizzyCount2,
873         });
874     }
875     catch (FirebaseException)
876     {
877         username = (await firebaseClient
878         .Child("users")
879         .Child(auth.GetUid())
880         .OnceSingleAsync<Users>()).username;
881
882         points2 = AppConstants.sixPoints;
883         await firebaseClient
884         .Child("FoodAndDrinkPoints")
885         .Child(auth.GetUid())
886         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, saveLeftOversCount = 1 }); ;
```

```
887     }
888     }
889     catch (NullReferenceException)
890     {
891         username = (await firebaseClient
892             .Child("users")
893             .Child(auth.GetUid())
894             .OnceSingleAsync<Users>()).username;
895
896         points2 = AppConstants.sixPoints;
897         await firebaseClient
898             .Child("FoodAndDrinkPoints")
899             .Child(auth.GetUid())
900             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, saveLeftOversCount = 1 });
901     }
902 }
903 /** This function updates the points in the FoodAndDrink category by four points.
It also increments the number of logs logged in the FoodAndDrink
904 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
905 */
906 public async void SteelStrawPoints()
907 {
908
909     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
910     auth = DependencyService.Get<IAuth>();
911
912     try
913     {
914         username = (await firebaseClient
915             .Child("users")
916             .Child(auth.GetUid())
917             .OnceSingleAsync<Users>()).username;
918
919         points2 = (await firebaseClient
920             .Child("FoodAndDrinkPoints")
921             .Child(auth.GetUid())
922             .OnceSingleAsync<FoodAndDrinkPoints>()).points;
923
924         points2 = points2 + AppConstants.fourPoints;
925
926         numberOfLogs2 = (await firebaseClient
927             .Child("FoodAndDrinkPoints")
928             .Child(auth.GetUid())
929             .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
930
931         numberOfLogs2++;
932
933         organicCount2 = (await firebaseClient
934             .Child("FoodAndDrinkPoints")
935             .Child(auth.GetUid())
936             .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
937
938         eatAllCount2 = (await firebaseClient
939             .Child("FoodAndDrinkPoints")
940             .Child(auth.GetUid())
941             .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
942
943         foodDeliverCount2 = (await firebaseClient
944             .Child("FoodAndDrinkPoints")
```

```
945         .Child(auth.GetUid())
946         .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
947
948         noMeatCount2 = (await firebaseClient
949         .Child("FoodAndDrinkPoints")
950         .Child(auth.GetUid())
951         .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
952
953         ownCoffeeCount2 = (await firebaseClient
954         .Child("FoodAndDrinkPoints")
955         .Child(auth.GetUid())
956         .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
957
958         reCoffeeMugCount2 = (await firebaseClient
959         .Child("FoodAndDrinkPoints")
960         .Child(auth.GetUid())
961         .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
962
963         saveLeftOversCount2 = (await firebaseClient
964         .Child("FoodAndDrinkPoints")
965         .Child(auth.GetUid())
966         .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
967
968         steelStrawCount2 = (await firebaseClient
969         .Child("FoodAndDrinkPoints")
970         .Child(auth.GetUid())
971         .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
972
973         steelStrawCount2++;
974
975         waterOverFizzyCount2 = (await firebaseClient
976         .Child("FoodAndDrinkPoints")
977         .Child(auth.GetUid())
978         .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
979
980         await firebaseClient
981         .Child("FoodAndDrinkPoints")
982         .Child(auth.GetUid())
983         .PutAsync(new FoodAndDrinkPoints()
984         {
985             username = username,
986             points = points2,
987             numberOfLogs = numberOfLogs2,
988             organicCount = organicCount2,
989             eatAllCount = eatAllCount2,
990             foodDeliverCount = foodDeliverCount2,
991             noMeatCount = noMeatCount2,
992             ownCoffeeCount = ownCoffeeCount2,
993             reCoffeeMugCount = reCoffeeMugCount2,
994             saveLeftOversCount = saveLeftOversCount2,
995             steelStrawCount = steelStrawCount2,
996             waterOverFizzyCount = waterOverFizzyCount2,
997         });
998     }
999     catch (FirebaseException)
1000     {
1001         username = (await firebaseClient
1002         .Child("users")
1003         .Child(auth.GetUid())
1004         .OnceSingleAsync<Users>()).username;
1005     }
```

```
1006         points2 = AppConstants.fourPoints;
1007         await firebaseClient
1008             .Child("FoodAndDrinkPoints")
1009             .Child(auth.GetUid())
1010             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, steelStrawCount = 1 }); ;
1011     }
1012     catch (NullReferenceException)
1013     {
1014         username = (await firebaseClient
1015             .Child("users")
1016             .Child(auth.GetUid())
1017             .OnceSingleAsync<Users>()).username;
1018
1019         points2 = AppConstants.fourPoints;
1020         await firebaseClient
1021             .Child("FoodAndDrinkPoints")
1022             .Child(auth.GetUid())
1023             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, steelStrawCount = 1 }); ;
1024     }
1025 }
1026
1027 /** This function updates the points in the FoodAndDrink category by six points.
It also increments the number of logs logged in the FoodAndDrink
1028 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1029 */
1030 public async void WaterOverFizzyPoints()
1031 {
1032
1033     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1034     auth = DependencyService.Get<IAuth>();
1035
1036     try
1037     {
1038         username = (await firebaseClient
1039             .Child("users")
1040             .Child(auth.GetUid())
1041             .OnceSingleAsync<Users>()).username;
1042
1043         points2 = (await firebaseClient
1044             .Child("FoodAndDrinkPoints")
1045             .Child(auth.GetUid())
1046             .OnceSingleAsync<FoodAndDrinkPoints>()).points;
1047
1048         points2 = points2 + AppConstants.sixPoints;
1049
1050         numberOfLogs2 = (await firebaseClient
1051             .Child("FoodAndDrinkPoints")
1052             .Child(auth.GetUid())
1053             .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
1054
1055         numberOfLogs2++;
1056
1057         organicCount2 = (await firebaseClient
1058             .Child("FoodAndDrinkPoints")
1059             .Child(auth.GetUid())
1060             .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
1061
1062         eatAllCount2 = (await firebaseClient
```

```
1063     .Child("FoodAndDrinkPoints")
1064     .Child(auth.GetUid())
1065     .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
1066
1067     foodDeliverCount2 = (await firebaseClient
1068     .Child("FoodAndDrinkPoints")
1069     .Child(auth.GetUid())
1070     .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
1071
1072     noMeatCount2 = (await firebaseClient
1073     .Child("FoodAndDrinkPoints")
1074     .Child(auth.GetUid())
1075     .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
1076
1077     ownCoffeeCount2 = (await firebaseClient
1078     .Child("FoodAndDrinkPoints")
1079     .Child(auth.GetUid())
1080     .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
1081
1082     reCoffeeMugCount2 = (await firebaseClient
1083     .Child("FoodAndDrinkPoints")
1084     .Child(auth.GetUid())
1085     .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
1086
1087     saveLeftOversCount2 = (await firebaseClient
1088     .Child("FoodAndDrinkPoints")
1089     .Child(auth.GetUid())
1090     .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
1091
1092     steelStrawCount2 = (await firebaseClient
1093     .Child("FoodAndDrinkPoints")
1094     .Child(auth.GetUid())
1095     .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
1096
1097     waterOverFizzyCount2 = (await firebaseClient
1098     .Child("FoodAndDrinkPoints")
1099     .Child(auth.GetUid())
1100     .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
1101
1102     waterOverFizzyCount2++;
1103
1104     await firebaseClient
1105     .Child("FoodAndDrinkPoints")
1106     .Child(auth.GetUid())
1107     .PutAsync(new FoodAndDrinkPoints()
1108     {
1109         username = username,
1110         points = points2,
1111         numberOfLogs = numberOfLogs2,
1112         organicCount = organicCount2,
1113         eatAllCount = eatAllCount2,
1114         foodDeliverCount = foodDeliverCount2,
1115         noMeatCount = noMeatCount2,
1116         ownCoffeeCount = ownCoffeeCount2,
1117         reCoffeeMugCount = reCoffeeMugCount2,
1118         saveLeftOversCount = saveLeftOversCount2,
1119         steelStrawCount = steelStrawCount2,
1120         waterOverFizzyCount = waterOverFizzyCount2,
1121     });
1122 }
1123 catch (FirebaseException)
```

```
1124     {
1125         username = (await firebaseClient
1126             .Child("users")
1127             .Child(auth.GetUid())
1128             .OnceSingleAsync<Users>()).username;
1129
1130         points2 = AppConstants.tenPoints;
1131         await firebaseClient
1132             .Child("FoodAndDrinkPoints")
1133             .Child(auth.GetUid())
1134             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, waterOverFizzyCount = 1 }); ;
1135     }
1136     catch (NullReferenceException)
1137     {
1138         username = (await firebaseClient
1139             .Child("users")
1140             .Child(auth.GetUid())
1141             .OnceSingleAsync<Users>()).username;
1142
1143         points2 = AppConstants.tenPoints;
1144         await firebaseClient
1145             .Child("FoodAndDrinkPoints")
1146             .Child(auth.GetUid())
1147             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, waterOverFizzyCount = 1 }); ;
1148     }
1149 }
1150 }
1151 }
1152 }
```

```
1  /!* \class The GetAchievementsData ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the GetAchievementsData ViewModel Class. It gets data that is
7  * needed for Achievements from firebase.
8  */
9  using System;
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using Xamarin.Forms;
14 namespace Application_Green_Quake.ViewModels
15 {
16     class GetAchievementsData
17     {
18         IAuth auth;
19         public static int fixCount = 0;
20         public static int wSShowerHeadCount = 0;
21         public static int waterOverFizzyCount = 0;
22         public static int createGroupCount = 0;
23         public static int communityCount = 0;
24         public static int donateCount = 0;
25         public static int groupCount = 0;
26         public static int shareCount = 0;
27         public static int awarenessCount = 0;
28         public static int fullDryerCount = 0;
29         public static int insulateWaterCount = 0;
30         public static int efficientThermostatCount = 0;
31         public static int isolateHomeCount = 0;
32         public static int ledLightBulbCount = 0;
33         public static int microwaveCount = 0;
34         public static int offSocketCount = 0;
35         public static int reBatteriesCount = 0;
36         public static int reBatCount = 0;
37         public static int fridgeCount = 0;
38         public static int draftSealCount = 0;
39         public static int ductSealCount = 0;
40         public static int solarPanelCount = 0;
41         public static int eatAllCount = 0;
42         public static int foodDeliverCount = 0;
43         public static int noMeatCount = 0;
44         public static int ownCoffeeCount = 0;
45         public static int reCoffeeMugCount = 0;
46         public static int saveLeftOversCount = 0;
47         public static int steelStrawCount = 0;
48         public static int brushingCount = 0;
49         public static int fullWasherCount = 0;
50         public static int showerCount = 0;
51         public static int timedShowerCount = 0;
52         public static int offLigtsCount = 0;
53         public static int matchesCount = 0;
54         public static int airOutCount = 0;
55         public static int nonHarmCount = 0;
56         public static int outsideCount = 0;
57         public static int plantIntoHomeCount = 0;
58         public static int toiletFlushCount = 0;
59         public static int campingCount = 0;
```



```
60     public static int picnicCount = 0;
61     public static int plantBushCount = 0;
62     public static int plantFlowerCount = 0;
63     public static int plantTreeCount = 0;
64     public static int scoopCount = 0;
65     public static int fruitGardenCount = 0;
66     public static int herbGardenCount = 0;
67     public static int vegetableGardenCount = 0;
68     public static int birdFeederCount = 0;
69     public static int clothNapkinCount = 0;
70     public static int clothTowelCount = 0;
71     public static int applianceCount = 0;
72     public static int productCount = 0;
73     public static int toothbrushCount = 0;
74     public static int clothesCount = 0;
75     public static int localCount = 0;
76     public static int looseLeafCount = 0;
77     public static int organicFoodCount = 0;
78     public static int reusableCount = 0;
79     public static int reBagCount = 0;
80     public static int carpoolCount = 0;
81     public static int cycleCount = 0;
82     public static int ecoCarCount = 0;
83     public static int transportCount = 0;
84     public static int walkCount = 0;
85     public static int billsCount = 0;
86     public static int compostCount = 0;
87     public static int setUpRecyclingBinCount = 0;
88     public static int bioBinBagsCount = 0;
89     public static int recyclingBinCount = 0;
90     public static int cisternCount = 0;
91     public static int rainBarrelCount = 0;
92     public static int reWaterCount = 0;
93     public static int showerBucketCount = 0;
94     public static int paperCount = 0;
95     public static int offElectronicsCount = 0;
96     public static int remoteWorkCount = 0;
97     public static int hangDryCount = 0;
98     public static int foodCount = 0;
99
100    /**
101     * This function sets the data that is needed for the Achievements Screen
102     */
103    public async void SetAchievementsData()
104    {
105        FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
106        auth = DependencyService.Get<IAuth>();
107
108        try
109        {
110            fixCount = (await firebaseClient
111                .Child("AdvancedPoints")
112                .Child(auth.GetUid())
113                .OnceSingleAsync<AdvancedPoints>()).fixCount;
114        }
115        catch (Exception e)
116        {
117            Console.Write(e);
118        }
119        try
```



```
120     {
121         createGroupCount = (await firebaseClient
122             .Child("CommunityPoints")
123             .Child(auth.GetUid())
124             .OnceSingleAsync<CommunityPoints>()).createGroupCount;
125     }
126     catch (Exception e)
127     {
128         Console.Write(e);
129     }
130     try
131     {
132         communityCount = (await firebaseClient
133             .Child("CommunityPoints")
134             .Child(auth.GetUid())
135             .OnceSingleAsync<CommunityPoints>()).communityCount;
136     }
137     catch (Exception e)
138     {
139         Console.Write(e);
140     }
141     try
142     {
143         donateCount = (await firebaseClient
144             .Child("CommunityPoints")
145             .Child(auth.GetUid())
146             .OnceSingleAsync<CommunityPoints>()).donateCount;
147     }
148     catch (Exception e)
149     {
150         Console.Write(e);
151     }
152     try
153     {
154         groupCount = (await firebaseClient
155             .Child("CommunityPoints")
156             .Child(auth.GetUid())
157             .OnceSingleAsync<CommunityPoints>()).groupCount;
158     }
159     catch (Exception e)
160     {
161         Console.Write(e);
162     }
163     try
164     {
165         shareCount = (await firebaseClient
166             .Child("CommunityPoints")
167             .Child(auth.GetUid())
168             .OnceSingleAsync<CommunityPoints>()).shareCount;
169     }
170     catch (Exception e)
171     {
172         Console.Write(e);
173     }
174     try
175     {
176         awarenessCount = (await firebaseClient
177             .Child("CommunityPoints")
178             .Child(auth.GetUid())
179             .OnceSingleAsync<CommunityPoints>()).awarenessCount;
180     }
```

```
181 catch (Exception e)
182 {
183     Console.WriteLine(e);
184 }
185 try
186 {
187     hangDryCount = (await firebaseClient
188         .Child("EnergyPoints")
189         .Child(auth.GetUid())
190         .OnceSingleAsync<EnergyPoints>()).hangDryCount;
191 }
192 catch (Exception e)
193 {
194     Console.WriteLine(e);
195 }
196 try
197 {
198     fullDryerCount = (await firebaseClient
199         .Child("EnergyPoints")
200         .Child(auth.GetUid())
201         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
202 }
203 catch (Exception e)
204 {
205     Console.WriteLine(e);
206 }
207 try
208 {
209     insulateWaterCount = (await firebaseClient
210         .Child("EnergyPoints")
211         .Child(auth.GetUid())
212         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
213 }
214 catch (Exception e)
215 {
216     Console.WriteLine(e);
217 }
218 try
219 {
220     efficientThermostatCount = (await firebaseClient
221         .Child("EnergyPoints")
222         .Child(auth.GetUid())
223         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
224 }
225 catch (Exception e)
226 {
227     Console.WriteLine(e);
228 }
229 try
230 {
231     isolateHomeCount = (await firebaseClient
232         .Child("EnergyPoints")
233         .Child(auth.GetUid())
234         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
235 }
236 catch (Exception e)
237 {
238     Console.WriteLine(e);
239 }
240 try
241 {
```

```
242         ledLightBulbCount = (await firebaseClient
243             .Child("EnergyPoints")
244             .Child(auth.GetUid())
245             .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
246     }
247     catch (Exception e)
248     {
249         Console.Write(e);
250     }
251     try
252     {
253         microwaveCount = (await firebaseClient
254             .Child("EnergyPoints")
255             .Child(auth.GetUid())
256             .OnceSingleAsync<EnergyPoints>()).microwaveCount;
257     }
258     catch (Exception e)
259     {
260         Console.Write(e);
261     }
262     try
263     {
264         offSocketCount = (await firebaseClient
265             .Child("EnergyPoints")
266             .Child(auth.GetUid())
267             .OnceSingleAsync<EnergyPoints>()).offSocketCount;
268     }
269     catch (Exception e)
270     {
271         Console.Write(e);
272     }
273     try
274     {
275         reBatteriesCount = (await firebaseClient
276             .Child("EnergyPoints")
277             .Child(auth.GetUid())
278             .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
279     }
280     catch (Exception e)
281     {
282         Console.Write(e);
283     }
284     try
285     {
286         reBatCount = (await firebaseClient
287             .Child("ShoppingPoints")
288             .Child(auth.GetUid())
289             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
290     }
291     catch (Exception e)
292     {
293         Console.Write(e);
294     }
295     try
296     {
297         fridgeCount = (await firebaseClient
298             .Child("EnergyPoints")
299             .Child(auth.GetUid())
300             .OnceSingleAsync<EnergyPoints>()).fridgeCount;
301     }
302     catch (Exception e)
```

```
303     {
304         Console.Write(e);
305     }
306     try
307     {
308         draftSealCount = (await firebaseClient
309             .Child("EnergyPoints")
310             .Child(auth.GetUid())
311             .OnceSingleAsync<EnergyPoints>()).draftSealCount;
312     }
313     catch (Exception e)
314     {
315         Console.Write(e);
316     }
317
318     try
319     {
320         ductSealCount = (await firebaseClient
321             .Child("EnergyPoints")
322             .Child(auth.GetUid())
323             .OnceSingleAsync<EnergyPoints>()).ductSealCount;
324     }
325     catch (Exception e)
326     {
327         Console.Write(e);
328     }
329     try
330     {
331         solarPanelCount = (await firebaseClient
332             .Child("EnergyPoints")
333             .Child(auth.GetUid())
334             .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
335     }
336     catch (Exception e)
337     {
338         Console.Write(e);
339     }
340     try
341     {
342         eatAllCount = (await firebaseClient
343             .Child("FoodAndDrinkPoints")
344             .Child(auth.GetUid())
345             .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
346     }
347     catch (Exception e)
348     {
349         Console.Write(e);
350     }
351     try
352     {
353         foodDeliverCount = (await firebaseClient
354             .Child("FoodAndDrinkPoints")
355             .Child(auth.GetUid())
356             .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
357     }
358     catch (Exception e)
359     {
360         Console.Write(e);
361     }
362     try
363     {
```

```
364         noMeatCount = (await firebaseClient
365             .Child("FoodAndDrinkPoints")
366             .Child(auth.GetUid())
367             .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
368     }
369     catch (Exception e)
370     {
371         Console.Write(e);
372     }
373     try
374     {
375         ownCoffeeCount = (await firebaseClient
376             .Child("FoodAndDrinkPoints")
377             .Child(auth.GetUid())
378             .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
379     }
380     catch (Exception e)
381     {
382         Console.Write(e);
383     }
384     try
385     {
386         reCoffeeMugCount = (await firebaseClient
387             .Child("FoodAndDrinkPoints")
388             .Child(auth.GetUid())
389             .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
390     }
391     catch (Exception e)
392     {
393         Console.Write(e);
394     }
395     try
396     {
397         saveLeftOversCount = (await firebaseClient
398             .Child("FoodAndDrinkPoints")
399             .Child(auth.GetUid())
400             .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
401     }
402     catch (Exception e)
403     {
404         Console.Write(e);
405     }
406     try
407     {
408         steelStrawCount = (await firebaseClient
409             .Child("FoodAndDrinkPoints")
410             .Child(auth.GetUid())
411             .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
412     }
413     catch (Exception e)
414     {
415         Console.Write(e);
416     }
417     try
418     {
419         waterOverFizzyCount = (await firebaseClient
420             .Child("FoodAndDrinkPoints")
421             .Child(auth.GetUid())
422             .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
423     }
424     catch (Exception e)
```

```
425     {
426         Console.Write(e);
427     }
428     try
429     {
430         brushingCount = (await firebaseClient
431             .Child("HabitsPoints")
432             .Child(auth.GetUid())
433             .OnceSingleAsync<HabitsPoints>()).brushingCount;
434     }
435     catch (Exception e)
436     {
437         Console.Write(e);
438     }
439     try
440     {
441         fullWasherCount = (await firebaseClient
442             .Child("HabitsPoints")
443             .Child(auth.GetUid())
444             .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
445     }
446     catch (Exception e)
447     {
448         Console.Write(e);
449     }
450     try
451     {
452         showerCount = (await firebaseClient
453             .Child("HabitsPoints")
454             .Child(auth.GetUid())
455             .OnceSingleAsync<HabitsPoints>()).showerCount;
456     }
457     catch (Exception e)
458     {
459         Console.Write(e);
460     }
461     try
462     {
463         timedShowerCount = (await firebaseClient
464             .Child("HabitsPoints")
465             .Child(auth.GetUid())
466             .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
467     }
468     catch (Exception e)
469     {
470         Console.Write(e);
471     }
472     try
473     {
474         offLigtsCount = (await firebaseClient
475             .Child("HabitsPoints")
476             .Child(auth.GetUid())
477             .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
478     }
479     catch (Exception e)
480     {
481         Console.Write(e);
482     }
483     try
484     {
485         matchesCount = (await firebaseClient
```

```
486         .Child("HabitsPoints")
487         .Child(auth.GetUid())
488         .OnceSingleAsync<HabitsPoints>()).matchesCount;
489     }
490     catch (Exception e)
491     {
492         Console.Write(e);
493     }
494     try
495     {
496         airOutCount = (await firebaseClient
497             .Child("HomePoints")
498             .Child(auth.GetUid())
499             .OnceSingleAsync<HomePoints>()).airOutCount;
500     }
501     catch (Exception e)
502     {
503         Console.Write(e);
504     }
505     try
506     {
507         nonHarmCount = (await firebaseClient
508             .Child("HomePoints")
509             .Child(auth.GetUid())
510             .OnceSingleAsync<HomePoints>()).nonHarmCount;
511     }
512     catch (Exception e)
513     {
514         Console.Write(e);
515     }
516     try
517     {
518         outsideCount = (await firebaseClient
519             .Child("HomePoints")
520             .Child(auth.GetUid())
521             .OnceSingleAsync<HomePoints>()).outsideCount;
522     }
523     catch (Exception e)
524     {
525         Console.Write(e);
526     }
527     try
528     {
529         plantIntoHomeCount = (await firebaseClient
530             .Child("HomePoints")
531             .Child(auth.GetUid())
532             .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;
533     }
534     catch (Exception e)
535     {
536         Console.Write(e);
537     }
538     try
539     {
540         toiletFlushCount = (await firebaseClient
541             .Child("HomePoints")
542             .Child(auth.GetUid())
543             .OnceSingleAsync<HomePoints>()).toiletFlushCount;
544     }
545     catch (Exception e)
546     {
```

```
547         Console.Write(e);
548     }
549     try
550     {
551         campingCount = (await firebaseClient
552             .Child("OutdoorsPoints")
553             .Child(auth.GetUid())
554             .OnceSingleAsync<OutdoorsPoints>()).campingCount;
555     }
556     catch (Exception e)
557     {
558         Console.Write(e);
559     }
560     try
561     {
562         picnicCount = (await firebaseClient
563             .Child("OutdoorsPoints")
564             .Child(auth.GetUid())
565             .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
566     }
567     catch (Exception e)
568     {
569         Console.Write(e);
570     }
571     try
572     {
573         plantBushCount = (await firebaseClient
574             .Child("OutdoorsPoints")
575             .Child(auth.GetUid())
576             .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
577     }
578     catch (Exception e)
579     {
580         Console.Write(e);
581     }
582     try
583     {
584         plantFlowerCount = (await firebaseClient
585             .Child("OutdoorsPoints")
586             .Child(auth.GetUid())
587             .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
588     }
589     catch (Exception e)
590     {
591         Console.Write(e);
592     }
593     try
594     {
595         plantTreeCount = (await firebaseClient
596             .Child("OutdoorsPoints")
597             .Child(auth.GetUid())
598             .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
599     }
600     catch (Exception e)
601     {
602         Console.Write(e);
603     }
604     try
605     {
606         scoopCount = (await firebaseClient
607             .Child("OutdoorsPoints")
```



```
608         .Child(auth.GetUid())
609         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
610     }
611     catch (Exception e)
612     {
613         Console.Write(e);
614     }
615     try
616     {
617         fruitGardenCount = (await firebaseClient
618             .Child("OutdoorsPoints")
619             .Child(auth.GetUid())
620             .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
621     }
622     catch (Exception e)
623     {
624         Console.Write(e);
625     }
626     try
627     {
628         herbGardenCount = (await firebaseClient
629             .Child("OutdoorsPoints")
630             .Child(auth.GetUid())
631             .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
632     }
633     catch (Exception e)
634     {
635         Console.Write(e);
636     }
637     try
638     {
639         vegetableGardenCount = (await firebaseClient
640             .Child("OutdoorsPoints")
641             .Child(auth.GetUid())
642             .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
643     }
644     catch (Exception e)
645     {
646         Console.Write(e);
647     }
648     try
649     {
650         birdFeederCount = (await firebaseClient
651             .Child("OutdoorsPoints")
652             .Child(auth.GetUid())
653             .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
654     }
655     catch (Exception e)
656     {
657         Console.Write(e);
658     }
659     try
660     {
661         clothNapkinCount = (await firebaseClient
662             .Child("ShoppingPoints")
663             .Child(auth.GetUid())
664             .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
665     }
666     catch (Exception e)
667     {
668         Console.Write(e);
```

```
669     }
670     try
671     {
672         clothTowelCount = (await firebaseClient
673             .Child("ShoppingPoints")
674             .Child(auth.GetUid())
675             .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
676     }
677     catch (Exception e)
678     {
679         Console.Write(e);
680     }
681     try
682     {
683         applianceCount = (await firebaseClient
684             .Child("ShoppingPoints")
685             .Child(auth.GetUid())
686             .OnceSingleAsync<ShoppingPoints>()).applianceCount;
687     }
688     catch (Exception e)
689     {
690         Console.Write(e);
691     }
692     try
693     {
694         productCount = (await firebaseClient
695             .Child("ShoppingPoints")
696             .Child(auth.GetUid())
697             .OnceSingleAsync<ShoppingPoints>()).productCount;
698     }
699     catch (Exception e)
700     {
701         Console.Write(e);
702     }
703     try
704     {
705         toothbrushCount = (await firebaseClient
706             .Child("ShoppingPoints")
707             .Child(auth.GetUid())
708             .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
709     }
710     catch (Exception e)
711     {
712         Console.Write(e);
713     }
714     try
715     {
716         clothesCount = (await firebaseClient
717             .Child("ShoppingPoints")
718             .Child(auth.GetUid())
719             .OnceSingleAsync<ShoppingPoints>()).clothesCount;
720     }
721     catch (Exception e)
722     {
723         Console.Write(e);
724     }
725     try
726     {
727         localCount = (await firebaseClient
728             .Child("ShoppingPoints")
729             .Child(auth.GetUid())
```

```
730         .OnceSingleAsync<ShoppingPoints>()).localCount;
731     }
732     catch (Exception e)
733     {
734         Console.Write(e);
735     }
736     try
737     {
738         looseLeafCount = (await firebaseClient
739             .Child("ShoppingPoints")
740             .Child(auth.GetUid())
741             .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
742     }
743     catch (Exception e)
744     {
745         Console.Write(e);
746     }
747     try
748     {
749         organicFoodCount = (await firebaseClient
750             .Child("ShoppingPoints")
751             .Child(auth.GetUid())
752             .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
753     }
754     catch (Exception e)
755     {
756         Console.Write(e);
757     }
758     try
759     {
760         reusableCount = (await firebaseClient
761             .Child("ShoppingPoints")
762             .Child(auth.GetUid())
763             .OnceSingleAsync<ShoppingPoints>()).reusableCount;
764     }
765     catch (Exception e)
766     {
767         Console.Write(e);
768     }
769     try
770     {
771         reBagCount = (await firebaseClient
772             .Child("ShoppingPoints")
773             .Child(auth.GetUid())
774             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
775     }
776     catch (Exception e)
777     {
778         Console.Write(e);
779     }
780     try
781     {
782         foodCount = (await firebaseClient
783             .Child("ShoppingPoints")
784             .Child(auth.GetUid())
785             .OnceSingleAsync<ShoppingPoints>()).foodCount;
786     }
787     catch (Exception e)
788     {
789         Console.Write(e);
790     }
```

```
791     try
792     {
793         carpoolCount = (await firebaseClient
794             .Child("TravelPoints")
795             .Child(auth.GetUid())
796             .OnceSingleAsync<TravelPoints>()).carpoolCount;
797     }
798     catch (Exception e)
799     {
800         Console.Write(e);
801     }
802     try
803     {
804         cycleCount = (await firebaseClient
805             .Child("TravelPoints")
806             .Child(auth.GetUid())
807             .OnceSingleAsync<TravelPoints>()).cycleCount;
808     }
809     catch (Exception e)
810     {
811         Console.Write(e);
812     }
813     try
814     {
815         ecoCarCount = (await firebaseClient
816             .Child("TravelPoints")
817             .Child(auth.GetUid())
818             .OnceSingleAsync<TravelPoints>()).ecoCarCount;
819     }
820     catch (Exception e)
821     {
822         Console.Write(e);
823     }
824     try
825     {
826         transportCount = (await firebaseClient
827             .Child("TravelPoints")
828             .Child(auth.GetUid())
829             .OnceSingleAsync<TravelPoints>()).transportCount;
830     }
831     catch (Exception e)
832     {
833         Console.Write(e);
834     }
835     try
836     {
837         walkCount = (await firebaseClient
838             .Child("TravelPoints")
839             .Child(auth.GetUid())
840             .OnceSingleAsync<TravelPoints>()).walkCount;
841     }
842     catch (Exception e)
843     {
844         Console.Write(e);
845     }
846     try
847     {
848         billsCount = (await firebaseClient
849             .Child("WastePoints")
850             .Child(auth.GetUid())
851             .OnceSingleAsync<WastePoints>()).billsCount;
```

```
852     }
853     catch (Exception e)
854     {
855         Console.WriteLine(e);
856     }
857     try
858     {
859         compostCount = (await firebaseClient
860             .Child("WastePoints")
861             .Child(auth.GetUid())
862             .OnceSingleAsync<WastePoints>()).compostCount;
863     }
864     catch (Exception e)
865     {
866         Console.WriteLine(e);
867     }
868     try
869     {
870         setUpRecyclingBinCount = (await firebaseClient
871             .Child("WastePoints")
872             .Child(auth.GetUid())
873             .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
874     }
875     catch (Exception e)
876     {
877         Console.WriteLine(e);
878     }
879     try
880     {
881         bioBinBagsCount = (await firebaseClient
882             .Child("WastePoints")
883             .Child(auth.GetUid())
884             .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
885     }
886     catch (Exception e)
887     {
888         Console.WriteLine(e);
889     }
890     try
891     {
892         recyclingBinCount = (await firebaseClient
893             .Child("WastePoints")
894             .Child(auth.GetUid())
895             .OnceSingleAsync<WastePoints>()).recyclingBinCount;
896     }
897     catch (Exception e)
898     {
899         Console.WriteLine(e);
900     }
901     try
902     {
903         cisternCount = (await firebaseClient
904             .Child("WaterPoints")
905             .Child(auth.GetUid())
906             .OnceSingleAsync<WaterPoints>()).cisternCount;
907     }
908     catch (Exception e)
909     {
910         Console.WriteLine(e);
911     }
912     try
```

```
913     {
914         rainBarrelCount = (await firebaseClient
915             .Child("WaterPoints")
916             .Child(auth.GetUid())
917             .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
918     }
919     catch (Exception e)
920     {
921         Console.Write(e);
922     }
923     try
924     {
925         reWaterCount = (await firebaseClient
926             .Child("WaterPoints")
927             .Child(auth.GetUid())
928             .OnceSingleAsync<WaterPoints>()).reWaterCount;
929     }
930     catch (Exception e)
931     {
932         Console.Write(e);
933     }
934     try
935     {
936         showerBucketCount = (await firebaseClient
937             .Child("WaterPoints")
938             .Child(auth.GetUid())
939             .OnceSingleAsync<WaterPoints>()).showerBucketCount;
940     }
941     catch (Exception e)
942     {
943         Console.Write(e);
944     }
945     try
946     {
947         wSShowerHeadCount = (await firebaseClient
948             .Child("WaterPoints")
949             .Child(auth.GetUid())
950             .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
951     }
952     catch (Exception e)
953     {
954         Console.Write(e);
955     }
956     try
957     {
958         paperCount = (await firebaseClient
959             .Child("WorkPoints")
960             .Child(auth.GetUid())
961             .OnceSingleAsync<WorkPoints>()).paperCount;
962     }
963     catch (Exception e)
964     {
965         Console.Write(e);
966     }
967     try
968     {
969         offElectronicsCount = (await firebaseClient
970             .Child("WorkPoints")
971             .Child(auth.GetUid())
972             .OnceSingleAsync<WorkPoints>()).offElectronicsCount;
973     }
```

```
974     catch (Exception e)
975     {
976         Console.WriteLine(e);
977     }
978     try
979     {
980         remoteWorkCount = (await firebaseClient
981             .Child("WorkPoints")
982             .Child(auth.GetUid())
983             .OnceSingleAsync<WorkPoints>()).remoteWorkCount;
984     }
985     catch (Exception e)
986     {
987         Console.WriteLine(e);
988     }
989 }
990 }
991 }
```

```
1 /!* \class The GetBadgeData ViewModel Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the GetBadgeData ViewModel Class. It gets data that is needed for
   badges from firebase.
7 */
8 using System;
9 using Application_Green_Quake.Models;
10 using Firebase.Database;
11 using Firebase.Database.Query;
12 using Xamarin.Forms;
13
14 namespace Application_Green_Quake.ViewModels
15 {
16     class GetBadgeData
17     {
18         IAuth auth;
19         public static int advancedLog = 0;
20         public static int habitsLog = 0;
21         public static int communityLog = 0;
22         public static int energyLog = 0;
23         public static int foodDrinkLog = 0;
24         public static int homeLog = 0;
25         public static int outdoorsLog = 0;
26         public static int shoppingLog = 0;
27         public static int travelLog = 0;
28         public static int wasteLog = 0;
29         public static int waterLog = 0;
30         public static int workLog = 0;
31
32         /**
33          * This function sets the data that is needed for the badges screen
34          */
35         public async void SetBadgeData()
36         {
37             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtdb.firebaseio.com/");
38             auth = DependencyService.Get<IAuth>();
39
40             try
41             {
42                 advancedLog = (await firebaseClient
43                     .Child("AdvancedPoints")
44                     .Child(auth.GetUid())
45                     .OnceSingleAsync<AdvancedPoints>()).numberOfLogs;
46             }
47             catch (Exception e)
48             {
49                 Console.Write(e);
50             }
51             try
52             {
53                 habitsLog = (await firebaseClient
54                     .Child("HabitsPoints")
55                     .Child(auth.GetUid())
56                     .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
57             }
58             catch (Exception e)
```



```
59     {
60         Console.Write(e);
61     }
62     try
63     {
64         communityLog = (await firebaseClient
65             .Child("CommunityPoints")
66             .Child(auth.GetUid())
67             .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
68     }
69     catch (Exception e)
70     {
71         Console.Write(e);
72     }
73     try
74     {
75         energyLog = (await firebaseClient
76             .Child("EnergyPoints")
77             .Child(auth.GetUid())
78             .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
79     }
80     catch (Exception e)
81     {
82         Console.Write(e);
83     }
84     try
85     {
86         foodDrinkLog = (await firebaseClient
87             .Child("FoodAndDrinkPoints")
88             .Child(auth.GetUid())
89             .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
90     }
91     catch (Exception e)
92     {
93         Console.Write(e);
94     }
95     try
96     {
97         homeLog = (await firebaseClient
98             .Child("HomePoints")
99             .Child(auth.GetUid())
100            .OnceSingleAsync<HomePoints>()).numberOfLogs;
101    }
102    catch (Exception e)
103    {
104        Console.Write(e);
105    }
106    try
107    {
108        outdoorsLog = (await firebaseClient
109            .Child("OutdoorsPoints")
110            .Child(auth.GetUid())
111            .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
112    }
113    catch (Exception e)
114    {
115        Console.Write(e);
116    }
117    try
118    {
119        shoppingLog = (await firebaseClient
```

```
120         .Child("ShoppingPoints")
121         .Child(auth.GetUid())
122         .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
123     }
124     catch (Exception e)
125     {
126         Console.Write(e);
127     }
128     try
129     {
130         travelLog = (await firebaseClient
131         .Child("TravelPoints")
132         .Child(auth.GetUid())
133         .OnceSingleAsync<TravelPoints>()).numberOfLogs;
134     }
135     catch (Exception e)
136     {
137         Console.Write(e);
138     }
139     try
140     {
141         wasteLog = (await firebaseClient
142         .Child("WastePoints")
143         .Child(auth.GetUid())
144         .OnceSingleAsync<WastePoints>()).numberOfLogs;
145     }
146     catch (Exception e)
147     {
148         Console.Write(e);
149     }
150     try
151     {
152         waterLog = (await firebaseClient
153         .Child("WaterPoints")
154         .Child(auth.GetUid())
155         .OnceSingleAsync<WaterPoints>()).numberOfLogs;
156     }
157     catch (Exception e)
158     {
159         Console.Write(e);
160     }
161     try
162     {
163         workLog = (await firebaseClient
164         .Child("WorkPoints")
165         .Child(auth.GetUid())
166         .OnceSingleAsync<WorkPoints>()).numberOfLogs;
167     }
168     catch (Exception e)
169     {
170         Console.Write(e);
171     }
172 }
173 }
174 }
```

```
1 /!* \class The GetData ViewModel Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the GetData ViewModel Class. It gets data that is needed for the
   applications back end and front end.
7 */
8 using Application_Green_Quake.Models;
9 using Firebase.Database;
10 using Firebase.Database.Query;
11 using System;
12 using Xamarin.Forms;
13
14 namespace Application_Green_Quake.ViewModels
15 {
16     class GetData
17     {
18         IAuth auth;
19         public static string username = "";
20         public static string bio = "";
21         public static int points = 0;
22         public static int lvl = 0;
23         public static string nation = "";
24
25         /**
26          * 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
27          * user. These are saved in global public variables that are used across the
   application for front end.
28          */
29         public async void SetData()
30         {
31             try
32             {
33                 FirebaseClient firebaseClient = new FirebaseClient("https://application-
   green-quake-default-rtdb.firebaseio.com/");
34                 auth = DependencyService.Get<IAuth>();
35
36                 username = (await firebaseClient
37                     .Child("users")
38                     .Child(auth.GetUid())
39                     .OnceSingleAsync<Users>()).username;
40
41                 bio = (await firebaseClient
42                     .Child("users")
43                     .Child(auth.GetUid())
44                     .OnceSingleAsync<Users>()).bio;
45
46                 nation = (await firebaseClient
47                     .Child("users")
48                     .Child(auth.GetUid())
49                     .OnceSingleAsync<Users>()).nation;
50
51                 points = (await firebaseClient
52                     .Child("Points")
53                     .Child(auth.GetUid())
54                     .OnceSingleAsync<Points>()).points;
55
56                 lvl = (int) Math.Floor((float) points / 10);
57             }
58         }
59     }
60 }
```

```
58         catch (Exception e)
59         {
60             Console.WriteLine(e);
61         }
62     }
63     /**
64     * This function just sets the level of the user in a global static variable shared
65     across all the classes to be displayed for the front end.
66     */
67     public async void SetLvl()
68     {
69         try
70         {
71             FirebaseClient firebaseClient =
72             new FirebaseClient("https://application-green-quake-default-
73             rtdb.firebaseio.com/");
74             auth = DependencyService.Get<IAuth>();
75             points = (await firebaseClient
76                 .Child("Points")
77                 .Child(auth.GetUid())
78                 .OnceSingleAsync<Points>()).points;
79             lvl = (int) Math.Floor((float) points / 10);
80         }
81         catch (Exception e)
82         {
83             Console.WriteLine(e);
84         }
85     }
86 }
87 }
```

```
1  /*! \class The HabitsPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the HabitsPointsUpdate ViewModel Class. It updates the data for the
   Habits Category of the application. The functions in this class
7  * work by reading in all the chosen data and updating the selected fields and then sending
   this data to back firebase.
8  *
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class HabitsPointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int brushingCount2 = 0;
23         int fullWasherCount2 = 0;
24         int showerCount2 = 0;
25         int timedShowerCount2 = 0;
26         int offLigtsCount2 = 0;
27         int matchesCount2 = 0;
28
29         string username = "";
30
31         IAuth auth;
32         /** This function updates the points in the Habits category by two points. It also
   increments the number of logs logged in the Habits
33         * category by one and increments the number of times this particular action was
   logged by one and sends this data to Firebase.
34         */
35         public async void BrushingPoints()
36         {
37
38             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtdb.firebaseio.com/");
39             auth = DependencyService.Get<IAuth>();
40
41             try
42             {
43                 username = (await firebaseClient
44                     .Child("users")
45                     .Child(auth.GetUid())
46                     .OnceSingleAsync<Users>()).username;
47
48                 points2 = (await firebaseClient
49                     .Child("HabitsPoints")
50                     .Child(auth.GetUid())
51                     .OnceSingleAsync<HabitsPoints>()).points;
52
53                 points2 = points2 + AppConstants.twoPoints;
54
55                 numberOfLogs2 = (await firebaseClient
56                     .Child("HabitsPoints")
```

```
57     .Child(auth.GetUid())
58     .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
59
60     numberOfLogs2++;
61
62     brushingCount2 = (await firebaseClient
63     .Child("HabitsPoints")
64     .Child(auth.GetUid())
65     .OnceSingleAsync<HabitsPoints>()).brushingCount;
66
67     brushingCount2++;
68
69     fullWasherCount2 = (await firebaseClient
70     .Child("HabitsPoints")
71     .Child(auth.GetUid())
72     .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
73
74     showerCount2 = (await firebaseClient
75     .Child("HabitsPoints")
76     .Child(auth.GetUid())
77     .OnceSingleAsync<HabitsPoints>()).showerCount;
78
79     timedShowerCount2 = (await firebaseClient
80     .Child("HabitsPoints")
81     .Child(auth.GetUid())
82     .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
83
84     offLigtsCount2 = (await firebaseClient
85     .Child("HabitsPoints")
86     .Child(auth.GetUid())
87     .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
88
89     matchesCount2 = (await firebaseClient
90     .Child("HabitsPoints")
91     .Child(auth.GetUid())
92     .OnceSingleAsync<HabitsPoints>()).matchesCount;
93
94     await firebaseClient
95     .Child("HabitsPoints")
96     .Child(auth.GetUid())
97     .PutAsync(new HabitsPoints()
98     {
99         username = username,
100        points = points2,
101        numberOfLogs = numberOfLogs2,
102        brushingCount = brushingCount2,
103        fullWasherCount = fullWasherCount2,
104        showerCount = showerCount2,
105        timedShowerCount = timedShowerCount2,
106        offLigtsCount = offLigtsCount2,
107        matchesCount = matchesCount2,
108    });
109 }
110 catch (FirebaseException)
111 {
112     username = (await firebaseClient
113     .Child("users")
114     .Child(auth.GetUid())
115     .OnceSingleAsync<Users>()).username;
116
117     points2 = AppConstants.twoPoints;
```

```
118         await firebaseClient
119             .Child("HabitsPoints")
120             .Child(auth.GetUid())
121             .PutAsync(new HabitsPoints() { username = username, points = points2,
numberOfLogs = 1, brushingCount = 1 }); ;
122     }
123 }
124 catch (NullReferenceException)
125 {
126     username = (await firebaseClient
127         .Child("users")
128         .Child(auth.GetUid())
129         .OnceSingleAsync<Users>()).username;
130
131     points2 = AppConstants.twoPoints;
132     await firebaseClient
133         .Child("HabitsPoints")
134         .Child(auth.GetUid())
135         .PutAsync(new HabitsPoints() { username = username, points = points2,
numberOfLogs = 1, brushingCount = 1 });
136     }
137 }
138 /** This function updates the points in the Habits category by eight points. It
also increments the number of logs logged in the Habits
139 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
140 */
141 public async void DishWasherFullPoints()
142 {
143
144     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
145     auth = DependencyService.Get<IAuth>();
146
147     try
148     {
149         username = (await firebaseClient
150             .Child("users")
151             .Child(auth.GetUid())
152             .OnceSingleAsync<Users>()).username;
153
154         points2 = (await firebaseClient
155             .Child("HabitsPoints")
156             .Child(auth.GetUid())
157             .OnceSingleAsync<HabitsPoints>()).points;
158
159         points2 = points2 + AppConstants.eightPoints;
160
161         numberOfLogs2 = (await firebaseClient
162             .Child("HabitsPoints")
163             .Child(auth.GetUid())
164             .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
165
166         numberOfLogs2++;
167
168         brushingCount2 = (await firebaseClient
169             .Child("HabitsPoints")
170             .Child(auth.GetUid())
171             .OnceSingleAsync<HabitsPoints>()).brushingCount;
172
173         fullWasherCount2 = (await firebaseClient
174             .Child("HabitsPoints")
```

```

175         .Child(auth.GetUid())
176         .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
177
178         fullWasherCount2++;
179
180         showerCount2 = (await firebaseClient
181         .Child("HabitsPoints")
182         .Child(auth.GetUid())
183         .OnceSingleAsync<HabitsPoints>()).showerCount;
184
185         timedShowerCount2 = (await firebaseClient
186         .Child("HabitsPoints")
187         .Child(auth.GetUid())
188         .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
189
190         offLigtsCount2 = (await firebaseClient
191         .Child("HabitsPoints")
192         .Child(auth.GetUid())
193         .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
194
195         matchesCount2 = (await firebaseClient
196         .Child("HabitsPoints")
197         .Child(auth.GetUid())
198         .OnceSingleAsync<HabitsPoints>()).matchesCount;
199
200         await firebaseClient
201         .Child("HabitsPoints")
202         .Child(auth.GetUid())
203         .PutAsync(new HabitsPoints()
204         {
205             username = username,
206             points = points2,
207             numberOfLogs = numberOfLogs2,
208             brushingCount = brushingCount2,
209             fullWasherCount = fullWasherCount2,
210             showerCount = showerCount2,
211             timedShowerCount = timedShowerCount2,
212             offLigtsCount = offLigtsCount2,
213             matchesCount = matchesCount2,
214         });
215     }
216     catch (FirebaseException)
217     {
218         username = (await firebaseClient
219         .Child("users")
220         .Child(auth.GetUid())
221         .OnceSingleAsync<Users>()).username;
222
223         points2 = AppConstants.eightPoints;
224         await firebaseClient
225         .Child("HabitsPoints")
226         .Child(auth.GetUid())
227         .PutAsync(new HabitsPoints() { username = username, points = points2,
228 numberOfLogs = 1, fullWasherCount = 1 }); ;
229     }
230     catch (NullReferenceException)
231     {
232         username = (await firebaseClient
233         .Child("users")
234         .Child(auth.GetUid())

```



```
235         .OnceSingleAsync<Users>()).username;
236
237         points2 = AppConstants.eightPoints;
238         await firebaseClient
239             .Child("HabitsPoints")
240             .Child(auth.GetUid())
241             .PutAsync(new HabitsPoints() { username = username, points = points2,
numberOfLogs = 1, fullWasherCount = 1 });
242     }
243 }
244 /** This function updates the points in the Habits category by six points. It also
increments the number of logs logged in the Habits
245 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
246 */
247 public async void ShowerInsteadPoints()
248 {
249
250     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
251     auth = DependencyService.Get<IAuth>();
252
253     try
254     {
255         username = (await firebaseClient
256             .Child("users")
257             .Child(auth.GetUid())
258             .OnceSingleAsync<Users>()).username;
259
260         points2 = (await firebaseClient
261             .Child("HabitsPoints")
262             .Child(auth.GetUid())
263             .OnceSingleAsync<HabitsPoints>()).points;
264
265         points2 = points2 + AppConstants.sixPoints;
266
267         numberOfLogs2 = (await firebaseClient
268             .Child("HabitsPoints")
269             .Child(auth.GetUid())
270             .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
271
272         numberOfLogs2++;
273
274         brushingCount2 = (await firebaseClient
275             .Child("HabitsPoints")
276             .Child(auth.GetUid())
277             .OnceSingleAsync<HabitsPoints>()).brushingCount;
278
279         fullWasherCount2 = (await firebaseClient
280             .Child("HabitsPoints")
281             .Child(auth.GetUid())
282             .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
283
284         showerCount2 = (await firebaseClient
285             .Child("HabitsPoints")
286             .Child(auth.GetUid())
287             .OnceSingleAsync<HabitsPoints>()).showerCount;
288
289         showerCount2++;
290
291         timedShowerCount2 = (await firebaseClient
292             .Child("HabitsPoints")
```

```

293         .Child(auth.GetUid())
294         .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
295
296         offLigtsCount2 = (await firebaseClient
297         .Child("HabitsPoints")
298         .Child(auth.GetUid())
299         .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
300
301         matchesCount2 = (await firebaseClient
302         .Child("HabitsPoints")
303         .Child(auth.GetUid())
304         .OnceSingleAsync<HabitsPoints>()).matchesCount;
305
306         await firebaseClient
307         .Child("HabitsPoints")
308         .Child(auth.GetUid())
309         .PutAsync(new HabitsPoints()
310         {
311             username = username,
312             points = points2,
313             numberOfLogs = numberOfLogs2,
314             brushingCount = brushingCount2,
315             fullWasherCount = fullWasherCount2,
316             showerCount = showerCount2,
317             timedShowerCount = timedShowerCount2,
318             offLigtsCount = offLigtsCount2,
319             matchesCount = matchesCount2,
320         });
321     }
322     catch (FirebaseException)
323     {
324         username = (await firebaseClient
325         .Child("users")
326         .Child(auth.GetUid())
327         .OnceSingleAsync<Users>()).username;
328
329         points2 = AppConstants.sixPoints;
330         await firebaseClient
331         .Child("HabitsPoints")
332         .Child(auth.GetUid())
333         .PutAsync(new HabitsPoints() { username = username, points = points2,
334         numberOfLogs = 1, showerCount = 1 }); ;
335     }
336     catch (NullReferenceException)
337     {
338         username = (await firebaseClient
339         .Child("users")
340         .Child(auth.GetUid())
341         .OnceSingleAsync<Users>()).username;
342
343         points2 = AppConstants.sixPoints;
344         await firebaseClient
345         .Child("HabitsPoints")
346         .Child(auth.GetUid())
347         .PutAsync(new HabitsPoints() { username = username, points = points2,
348         numberOfLogs = 1, showerCount = 1 });
349     }
350     /** This function updates the points in the Habits category by four points. It also
351     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.
352     */
353     public async void TimedShowerInsteadPoints()
354     {
355
356         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
357         auth = DependencyService.Get<IAuth>();
358
359         try
360         {
361             username = (await firebaseClient
362                 .Child("users")
363                 .Child(auth.GetUid())
364                 .OnceSingleAsync<Users>()).username;
365
366             points2 = (await firebaseClient
367                 .Child("HabitsPoints")
368                 .Child(auth.GetUid())
369                 .OnceSingleAsync<HabitsPoints>()).points;
370
371             points2 = points2 + AppConstants.fourPoints;
372
373             numberOfLogs2 = (await firebaseClient
374                 .Child("HabitsPoints")
375                 .Child(auth.GetUid())
376                 .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
377
378             numberOfLogs2++;
379
380             brushingCount2 = (await firebaseClient
381                 .Child("HabitsPoints")
382                 .Child(auth.GetUid())
383                 .OnceSingleAsync<HabitsPoints>()).brushingCount;
384
385             fullWasherCount2 = (await firebaseClient
386                 .Child("HabitsPoints")
387                 .Child(auth.GetUid())
388                 .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
389
390             showerCount2 = (await firebaseClient
391                 .Child("HabitsPoints")
392                 .Child(auth.GetUid())
393                 .OnceSingleAsync<HabitsPoints>()).showerCount;
394
395             timedShowerCount2 = (await firebaseClient
396                 .Child("HabitsPoints")
397                 .Child(auth.GetUid())
398                 .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
399
400             timedShowerCount2++;
401
402             offLigtsCount2 = (await firebaseClient
403                 .Child("HabitsPoints")
404                 .Child(auth.GetUid())
405                 .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
406
407             matchesCount2 = (await firebaseClient
408                 .Child("HabitsPoints")
409                 .Child(auth.GetUid())
410                 .OnceSingleAsync<HabitsPoints>()).matchesCount;
```

```

411
412     await firebaseClient
413         .Child("HabitsPoints")
414         .Child(auth.GetUid())
415         .PutAsync(new HabitsPoints()
416             {
417                 username = username,
418                 points = points2,
419                 numberOfLogs = numberOfLogs2,
420                 brushingCount = brushingCount2,
421                 fullWasherCount = fullWasherCount2,
422                 showerCount = showerCount2,
423                 timedShowerCount = timedShowerCount2,
424                 offLightsCount = offLightsCount2,
425                 matchesCount = matchesCount2,
426             });
427     }
428     catch (FirebaseException)
429     {
430         username = (await firebaseClient
431             .Child("users")
432             .Child(auth.GetUid())
433             .OnceSingleAsync<Users>()).username;
434
435         points2 = AppConstants.fourPoints;
436         await firebaseClient
437             .Child("HabitsPoints")
438             .Child(auth.GetUid())
439             .PutAsync(new HabitsPoints() { username = username, points = points2,
440 numberOfLogs = 1, timedShowerCount = 1 }); ;
441     }
442     catch (NullReferenceException)
443     {
444         username = (await firebaseClient
445             .Child("users")
446             .Child(auth.GetUid())
447             .OnceSingleAsync<Users>()).username;
448
449         points2 = AppConstants.fourPoints;
450         await firebaseClient
451             .Child("HabitsPoints")
452             .Child(auth.GetUid())
453             .PutAsync(new HabitsPoints() { username = username, points = points2,
454 numberOfLogs = 1, timedShowerCount = 1 }); ;
455     }
456     /** This function updates the points in the Habits category by two points. It also
457 increments the number of logs logged in the Habits
458 * category by one and increments the number of times this particular action was
459 logged by one and sends this data to Firebase.
460 */
461     public async void OffLightsPoints()
462     {
463         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
464 quake-default-rtdb.firebaseio.com/");
465         auth = DependencyService.Get<IAuth>();
466
467         try
468         {
469             username = (await firebaseClient

```

```
468     .Child("users")
469     .Child(auth.GetUid())
470     .OnceSingleAsync<Users>()).username;
471
472     points2 = (await firebaseClient
473     .Child("HabitsPoints")
474     .Child(auth.GetUid())
475     .OnceSingleAsync<HabitsPoints>()).points;
476
477     points2 = points2 + AppConstants.twoPoints;
478
479     numberOfLogs2 = (await firebaseClient
480     .Child("HabitsPoints")
481     .Child(auth.GetUid())
482     .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
483
484     numberOfLogs2++;
485
486     brushingCount2 = (await firebaseClient
487     .Child("HabitsPoints")
488     .Child(auth.GetUid())
489     .OnceSingleAsync<HabitsPoints>()).brushingCount;
490
491     fullWasherCount2 = (await firebaseClient
492     .Child("HabitsPoints")
493     .Child(auth.GetUid())
494     .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
495
496     showerCount2 = (await firebaseClient
497     .Child("HabitsPoints")
498     .Child(auth.GetUid())
499     .OnceSingleAsync<HabitsPoints>()).showerCount;
500
501     timedShowerCount2 = (await firebaseClient
502     .Child("HabitsPoints")
503     .Child(auth.GetUid())
504     .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
505
506     offLigtsCount2 = (await firebaseClient
507     .Child("HabitsPoints")
508     .Child(auth.GetUid())
509     .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
510
511     offLigtsCount2++;
512
513     matchesCount2 = (await firebaseClient
514     .Child("HabitsPoints")
515     .Child(auth.GetUid())
516     .OnceSingleAsync<HabitsPoints>()).matchesCount;
517
518     await firebaseClient
519     .Child("HabitsPoints")
520     .Child(auth.GetUid())
521     .PutAsync(new HabitsPoints()
522     {
523         username = username,
524         points = points2,
525         numberOfLogs = numberOfLogs2,
526         brushingCount = brushingCount2,
527         fullWasherCount = fullWasherCount2,
528         showerCount = showerCount2,
```

```

529         timedShowerCount = timedShowerCount2,
530         offLightsCount = offLightsCount2,
531         matchesCount = matchesCount2,
532     });
533 }
534 catch (FirebaseException)
535 {
536     username = (await firebaseClient
537         .Child("users")
538         .Child(auth.GetUid())
539         .OnceSingleAsync<Users>()).username;
540
541     points2 = AppConstants.twoPoints;
542     await firebaseClient
543         .Child("HabitsPoints")
544         .Child(auth.GetUid())
545         .PutAsync(new HabitsPoints() { username = username, points = points2,
546 numberOfLogs = 1, offLightsCount = 1 }); ;
547     }
548 catch (NullReferenceException)
549 {
550     username = (await firebaseClient
551         .Child("users")
552         .Child(auth.GetUid())
553         .OnceSingleAsync<Users>()).username;
554
555     points2 = AppConstants.twoPoints;
556     await firebaseClient
557         .Child("HabitsPoints")
558         .Child(auth.GetUid())
559         .PutAsync(new HabitsPoints() { username = username, points = points2,
560 numberOfLogs = 1, offLightsCount = 1 });
561     }
562     /** This function updates the points in the Habits category by two points. It also
563 increments the number of logs logged in the Habits
564 * category by one and increments the number of times this particular action was
565 logged by one and sends this data to Firebase.
566 */
567 public async void MatchesPoints()
568 {
569     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
570 quake-default-rtdb.firebaseio.com/");
571     auth = DependencyService.Get<IAuth>();
572
573     try
574     {
575         username = (await firebaseClient
576             .Child("users")
577             .Child(auth.GetUid())
578             .OnceSingleAsync<Users>()).username;
579
580         points2 = (await firebaseClient
581             .Child("HabitsPoints")
582             .Child(auth.GetUid())
583             .OnceSingleAsync<HabitsPoints>()).points;
584
585         points2 = points2 + AppConstants.twoPoints;
586
587         numberOfLogs2 = (await firebaseClient

```

```
586     .Child("HabitsPoints")
587     .Child(auth.GetUid())
588     .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
589
590     numberOfLogs2++;
591
592     brushingCount2 = (await firebaseClient
593     .Child("HabitsPoints")
594     .Child(auth.GetUid())
595     .OnceSingleAsync<HabitsPoints>()).brushingCount;
596
597     fullWasherCount2 = (await firebaseClient
598     .Child("HabitsPoints")
599     .Child(auth.GetUid())
600     .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
601
602     showerCount2 = (await firebaseClient
603     .Child("HabitsPoints")
604     .Child(auth.GetUid())
605     .OnceSingleAsync<HabitsPoints>()).showerCount;
606
607     timedShowerCount2 = (await firebaseClient
608     .Child("HabitsPoints")
609     .Child(auth.GetUid())
610     .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
611
612     offLigtsCount2 = (await firebaseClient
613     .Child("HabitsPoints")
614     .Child(auth.GetUid())
615     .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
616
617     matchesCount2 = (await firebaseClient
618     .Child("HabitsPoints")
619     .Child(auth.GetUid())
620     .OnceSingleAsync<HabitsPoints>()).matchesCount;
621
622     matchesCount2++;
623
624     await firebaseClient
625     .Child("HabitsPoints")
626     .Child(auth.GetUid())
627     .PutAsync(new HabitsPoints()
628     {
629         username = username,
630         points = points2,
631         numberOfLogs = numberOfLogs2,
632         brushingCount = brushingCount2,
633         fullWasherCount = fullWasherCount2,
634         showerCount = showerCount2,
635         timedShowerCount = timedShowerCount2,
636         offLigtsCount = offLigtsCount2,
637         matchesCount = matchesCount2,
638     });
639 }
640 catch (FirebaseException)
641 {
642     username = (await firebaseClient
643     .Child("users")
644     .Child(auth.GetUid())
645     .OnceSingleAsync<Users>()).username;
646
```

```
647         points2 = AppConstants.twoPoints;
648         await firebaseClient
649             .Child("HabitsPoints")
650             .Child(auth.GetUid())
651             .PutAsync(new HabitsPoints() { username = username, points = points2,
numberOfLogs = 1, matchesCount = 1 }); ;
652     }
653     catch (NullReferenceException)
654     {
655         username = (await firebaseClient
656             .Child("users")
657             .Child(auth.GetUid())
658             .OnceSingleAsync<Users>()).username;
659
660         points2 = AppConstants.twoPoints;
661         await firebaseClient
662             .Child("HabitsPoints")
663             .Child(auth.GetUid())
664             .PutAsync(new HabitsPoints() { username = username, points = points2,
numberOfLogs = 1, matchesCount = 1 });
665     }
666 }
667 }
668 }
669 }
```



```
1  /*! \class The HabitsPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the HomePointsUpdate ViewModel Class. It updates the data for the
   Home Category of the application. The functions in this class
7  * work by reading in all the chosen data and updating the selected fields and then sending
   this data to back firebase.
8  *
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class HomePointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int airOutCount2 = 0;
23         int nonHarmCount2 = 0;
24         int outsideCount2 = 0;
25         int plantIntoHomeCount2 = 0;
26         int toiletFlushCount2 = 0;
27
28         string username = "";
29
30         IAuth auth;
31         /** This function updates the points in the Home category by two points. It also
   increments the number of logs logged in the Home
32         * category by one and increments the number of times this particular action was
   logged by one and sends this data to Firebase.
33         */
34         public async void AirOutPoints()
35         {
36
37             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtdb.firebaseio.com/");
38             auth = DependencyService.Get<IAuth>();
39
40             try
41             {
42                 username = (await firebaseClient
43                     .Child("users")
44                     .Child(auth.GetUid())
45                     .OnceSingleAsync<Users>()).username;
46
47                 points2 = (await firebaseClient
48                     .Child("HomePoints")
49                     .Child(auth.GetUid())
50                     .OnceSingleAsync<HomePoints>()).points;
51
52                 points2 = points2 + AppConstants.twoPoints;
53
54                 numberOfLogs2 = (await firebaseClient
55                     .Child("HomePoints")
56                     .Child(auth.GetUid())
```

```
57         .OnceSingleAsync<HomePoints>()).numberOfLogs;
58
59         numberOfLogs2++;
60
61         airOutCount2 = (await firebaseClient
62             .Child("HomePoints")
63             .Child(auth.GetUid())
64             .OnceSingleAsync<HomePoints>()).airOutCount;
65
66         airOutCount2++;
67
68         nonHarmCount2 = (await firebaseClient
69             .Child("HomePoints")
70             .Child(auth.GetUid())
71             .OnceSingleAsync<HomePoints>()).nonHarmCount;
72
73         outsideCount2 = (await firebaseClient
74             .Child("HomePoints")
75             .Child(auth.GetUid())
76             .OnceSingleAsync<HomePoints>()).outsideCount;
77
78         plantIntoHomeCount2 = (await firebaseClient
79             .Child("HomePoints")
80             .Child(auth.GetUid())
81             .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;
82
83         toiletFlushCount2 = (await firebaseClient
84             .Child("HomePoints")
85             .Child(auth.GetUid())
86             .OnceSingleAsync<HomePoints>()).toiletFlushCount;
87
88         await firebaseClient
89             .Child("HomePoints")
90             .Child(auth.GetUid())
91             .PutAsync(new HomePoints()
92             {
93                 username = username,
94                 points = points2,
95                 numberOfLogs = numberOfLogs2,
96                 airOutCount = airOutCount2,
97                 nonHarmCount = nonHarmCount2,
98                 outsideCount = outsideCount2,
99                 plantIntoHomeCount = plantIntoHomeCount2,
100                toiletFlushCount = toiletFlushCount2,
101            });
102     }
103     catch (FirebaseException)
104     {
105         username = (await firebaseClient
106             .Child("users")
107             .Child(auth.GetUid())
108             .OnceSingleAsync<Users>()).username;
109
110         points2 = AppConstants.twoPoints;
111         await firebaseClient
112             .Child("HomePoints")
113             .Child(auth.GetUid())
114             .PutAsync(new HomePoints() { username = username, points = points2,
115 numberOfLogs = 1, airOutCount = 1 }); ;
116     }
```

```
117     catch (NullReferenceException)
118     {
119         username = (await firebaseClient
120             .Child("users")
121             .Child(auth.GetUid())
122             .OnceSingleAsync<Users>()).username;
123
124         points2 = AppConstants.twoPoints;
125         await firebaseClient
126             .Child("HomePoints")
127             .Child(auth.GetUid())
128             .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, airOutCount = 1 });
129     }
130 }
131 /** This function updates the points in the Home category by four points. It also
increments the number of logs logged in the Home
132 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
133 */
134 public async void NonHarmfulPoints()
135 {
136
137     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
138     auth = DependencyService.Get<IAuth>();
139
140     try
141     {
142         username = (await firebaseClient
143             .Child("users")
144             .Child(auth.GetUid())
145             .OnceSingleAsync<Users>()).username;
146
147         points2 = (await firebaseClient
148             .Child("HomePoints")
149             .Child(auth.GetUid())
150             .OnceSingleAsync<HomePoints>()).points;
151
152         points2 = points2 + AppConstants.fourPoints;
153
154         numberOfLogs2 = (await firebaseClient
155             .Child("HomePoints")
156             .Child(auth.GetUid())
157             .OnceSingleAsync<HomePoints>()).numberOfLogs;
158
159         numberOfLogs2++;
160
161         airOutCount2 = (await firebaseClient
162             .Child("HomePoints")
163             .Child(auth.GetUid())
164             .OnceSingleAsync<HomePoints>()).airOutCount;
165
166         nonHarmCount2 = (await firebaseClient
167             .Child("HomePoints")
168             .Child(auth.GetUid())
169             .OnceSingleAsync<HomePoints>()).nonHarmCount;
170
171         nonHarmCount2++;
172
173         outsideCount2 = (await firebaseClient
174             .Child("HomePoints")
```

```

175         .Child(auth.GetUid())
176         .OnceSingleAsync<HomePoints>()).outsideCount;
177
178         plantIntoHomeCount2 = (await firebaseClient
179         .Child("HomePoints")
180         .Child(auth.GetUid())
181         .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;
182
183         toiletFlushCount2 = (await firebaseClient
184         .Child("HomePoints")
185         .Child(auth.GetUid())
186         .OnceSingleAsync<HomePoints>()).toiletFlushCount;
187
188         await firebaseClient
189         .Child("HomePoints")
190         .Child(auth.GetUid())
191         .PutAsync(new HomePoints()
192         {
193             username = username,
194             points = points2,
195             numberOfLogs = numberOfLogs2,
196             airOutCount = airOutCount2,
197             nonHarmCount = nonHarmCount2,
198             outsideCount = outsideCount2,
199             plantIntoHomeCount = plantIntoHomeCount2,
200             toiletFlushCount = toiletFlushCount2,
201         });
202     }
203     catch (FirebaseException)
204     {
205         username = (await firebaseClient
206         .Child("users")
207         .Child(auth.GetUid())
208         .OnceSingleAsync<Users>()).username;
209
210         points2 = AppConstants.fourPoints;
211         await firebaseClient
212         .Child("HomePoints")
213         .Child(auth.GetUid())
214         .PutAsync(new HomePoints() { username = username, points = points2,
215         numberOfLogs = 1, nonHarmCount = 1 }); ;
216     }
217     catch (NullReferenceException)
218     {
219         username = (await firebaseClient
220         .Child("users")
221         .Child(auth.GetUid())
222         .OnceSingleAsync<Users>()).username;
223
224         points2 = AppConstants.fourPoints;
225         await firebaseClient
226         .Child("HomePoints")
227         .Child(auth.GetUid())
228         .PutAsync(new HomePoints() { username = username, points = points2,
229         numberOfLogs = 1, nonHarmCount = 1 }); ;
230     }
231     /** This function updates the points in the Home category by two points. It also
232     increments the number of logs logged in the Home
233     * category by one and increments the number of times this particular action was
234     logged by one and sends this data to Firebase.

```

```
233     */
234     public async void OutsidePoints()
235     {
236
237         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
238         auth = DependencyService.Get<IAuth>();
239
240         try
241         {
242             username = (await firebaseClient
243                 .Child("users")
244                 .Child(auth.GetUid())
245                 .OnceSingleAsync<Users>()).username;
246
247             points2 = (await firebaseClient
248                 .Child("HomePoints")
249                 .Child(auth.GetUid())
250                 .OnceSingleAsync<HomePoints>()).points;
251
252             points2 = points2 + AppConstants.twoPoints;
253
254             numberOfLogs2 = (await firebaseClient
255                 .Child("HomePoints")
256                 .Child(auth.GetUid())
257                 .OnceSingleAsync<HomePoints>()).numberOfLogs;
258
259             numberOfLogs2++;
260
261             airOutCount2 = (await firebaseClient
262                 .Child("HomePoints")
263                 .Child(auth.GetUid())
264                 .OnceSingleAsync<HomePoints>()).airOutCount;
265
266             nonHarmCount2 = (await firebaseClient
267                 .Child("HomePoints")
268                 .Child(auth.GetUid())
269                 .OnceSingleAsync<HomePoints>()).nonHarmCount;
270
271             outsideCount2 = (await firebaseClient
272                 .Child("HomePoints")
273                 .Child(auth.GetUid())
274                 .OnceSingleAsync<HomePoints>()).outsideCount;
275
276             outsideCount2++;
277
278             plantIntoHomeCount2 = (await firebaseClient
279                 .Child("HomePoints")
280                 .Child(auth.GetUid())
281                 .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;
282
283             toiletFlushCount2 = (await firebaseClient
284                 .Child("HomePoints")
285                 .Child(auth.GetUid())
286                 .OnceSingleAsync<HomePoints>()).toiletFlushCount;
287
288             await firebaseClient
289                 .Child("HomePoints")
290                 .Child(auth.GetUid())
291                 .PutAsync(new HomePoints()
292                 {
```

```
293         username = username,
294         points = points2,
295         numberOfLogs = numberOfLogs2,
296         airOutCount = airOutCount2,
297         nonHarmCount = nonHarmCount2,
298         outsideCount = outsideCount2,
299         plantIntoHomeCount = plantIntoHomeCount2,
300         toiletFlushCount = toiletFlushCount2,
301     });
302 }
303 catch (FirebaseException)
304 {
305     username = (await firebaseClient
306     .Child("users")
307     .Child(auth.GetUid())
308     .OnceSingleAsync<Users>()).username;
309
310     points2 = AppConstants.twoPoints;
311     await firebaseClient
312     .Child("HomePoints")
313     .Child(auth.GetUid())
314     .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, outsideCount = 1 }); ;
315 }
316 }
317 catch (NullReferenceException)
318 {
319     username = (await firebaseClient
320     .Child("users")
321     .Child(auth.GetUid())
322     .OnceSingleAsync<Users>()).username;
323
324     points2 = AppConstants.twoPoints;
325     await firebaseClient
326     .Child("HomePoints")
327     .Child(auth.GetUid())
328     .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, outsideCount = 1 });
329 }
330 }
331 /** This function updates the points in the Home category by four points. It also
increments the number of logs logged in the Home
332 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
333 */
334 public async void PlantsInsidePoints()
335 {
336
337     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
338     auth = DependencyService.Get<IAuth>();
339
340     try
341     {
342         username = (await firebaseClient
343         .Child("users")
344         .Child(auth.GetUid())
345         .OnceSingleAsync<Users>()).username;
346
347         points2 = (await firebaseClient
348         .Child("HomePoints")
349         .Child(auth.GetUid())
```

```
350     .OnceSingleAsync<HomePoints>()).points;
351
352     points2 = points2 + AppConstants.fourPoints;
353
354     numberOfLogs2 = (await firebaseClient
355     .Child("HomePoints")
356     .Child(auth.GetUid())
357     .OnceSingleAsync<HomePoints>()).numberOfLogs;
358
359     numberOfLogs2++;
360
361     airOutCount2 = (await firebaseClient
362     .Child("HomePoints")
363     .Child(auth.GetUid())
364     .OnceSingleAsync<HomePoints>()).airOutCount;
365
366     nonHarmCount2 = (await firebaseClient
367     .Child("HomePoints")
368     .Child(auth.GetUid())
369     .OnceSingleAsync<HomePoints>()).nonHarmCount;
370
371     outsideCount2 = (await firebaseClient
372     .Child("HomePoints")
373     .Child(auth.GetUid())
374     .OnceSingleAsync<HomePoints>()).outsideCount;
375
376     plantIntoHomeCount2 = (await firebaseClient
377     .Child("HomePoints")
378     .Child(auth.GetUid())
379     .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;
380
381     plantIntoHomeCount2++;
382
383     toiletFlushCount2 = (await firebaseClient
384     .Child("HomePoints")
385     .Child(auth.GetUid())
386     .OnceSingleAsync<HomePoints>()).toiletFlushCount;
387
388     await firebaseClient
389     .Child("HomePoints")
390     .Child(auth.GetUid())
391     .PutAsync(new HomePoints()
392     {
393         username = username,
394         points = points2,
395         numberOfLogs = numberOfLogs2,
396         airOutCount = airOutCount2,
397         nonHarmCount = nonHarmCount2,
398         outsideCount = outsideCount2,
399         plantIntoHomeCount = plantIntoHomeCount2,
400         toiletFlushCount = toiletFlushCount2,
401     });
402 }
403 catch (FirebaseException)
404 {
405     username = (await firebaseClient
406     .Child("users")
407     .Child(auth.GetUid())
408     .OnceSingleAsync<Users>()).username;
409
410     points2 = AppConstants.fourPoints;
```

```

411         await firebaseClient
412             .Child("HomePoints")
413             .Child(auth.GetUid())
414             .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, plantIntoHomeCount = 1 }); ;
415     }
416 }
417 catch (NullReferenceException)
418 {
419     username = (await firebaseClient
420         .Child("users")
421         .Child(auth.GetUid())
422         .OnceSingleAsync<Users>()).username;
423
424     points2 = AppConstants.fourPoints;
425     await firebaseClient
426         .Child("HomePoints")
427         .Child(auth.GetUid())
428         .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, plantIntoHomeCount = 1 }); ;
429     }
430 }
431 /** This function updates the points in the Home category by four points. It also
increments the number of logs logged in the Home
432 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
433 */
434 public async void ToiletPoints()
435 {
436
437     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
438     auth = DependencyService.Get<IAuth>();
439
440     try
441     {
442         username = (await firebaseClient
443             .Child("users")
444             .Child(auth.GetUid())
445             .OnceSingleAsync<Users>()).username;
446
447         points2 = (await firebaseClient
448             .Child("HomePoints")
449             .Child(auth.GetUid())
450             .OnceSingleAsync<HomePoints>()).points;
451
452         points2 = points2 + AppConstants.fourPoints;
453
454         numberOfLogs2 = (await firebaseClient
455             .Child("HomePoints")
456             .Child(auth.GetUid())
457             .OnceSingleAsync<HomePoints>()).numberOfLogs;
458
459         numberOfLogs2++;
460
461         airOutCount2 = (await firebaseClient
462             .Child("HomePoints")
463             .Child(auth.GetUid())
464             .OnceSingleAsync<HomePoints>()).airOutCount;
465
466         nonHarmCount2 = (await firebaseClient
467             .Child("HomePoints")

```



```
468         .Child(auth.GetUid())
469         .OnceSingleAsync<HomePoints>()).nonHarmCount;
470
471         outsideCount2 = (await firebaseClient
472         .Child("HomePoints")
473         .Child(auth.GetUid())
474         .OnceSingleAsync<HomePoints>()).outsideCount;
475
476         plantIntoHomeCount2 = (await firebaseClient
477         .Child("HomePoints")
478         .Child(auth.GetUid())
479         .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;
480
481         toiletFlushCount2 = (await firebaseClient
482         .Child("HomePoints")
483         .Child(auth.GetUid())
484         .OnceSingleAsync<HomePoints>()).toiletFlushCount;
485
486         toiletFlushCount2++;
487
488         await firebaseClient
489         .Child("HomePoints")
490         .Child(auth.GetUid())
491         .PutAsync(new HomePoints()
492         {
493             username = username,
494             points = points2,
495             numberOfLogs = numberOfLogs2,
496             airOutCount = airOutCount2,
497             nonHarmCount = nonHarmCount2,
498             outsideCount = outsideCount2,
499             plantIntoHomeCount = plantIntoHomeCount2,
500             toiletFlushCount = toiletFlushCount2,
501         });
502     }
503     catch (FirebaseException)
504     {
505         username = (await firebaseClient
506         .Child("users")
507         .Child(auth.GetUid())
508         .OnceSingleAsync<Users>()).username;
509
510         points2 = AppConstants.fourPoints;
511         await firebaseClient
512         .Child("HomePoints")
513         .Child(auth.GetUid())
514         .PutAsync(new HomePoints() { username = username, points = points2,
515         numberOfLogs = 1, toiletFlushCount = 1 }); ;
516     }
517     catch (NullReferenceException)
518     {
519         username = (await firebaseClient
520         .Child("users")
521         .Child(auth.GetUid())
522         .OnceSingleAsync<Users>()).username;
523
524         points2 = AppConstants.fourPoints;
525         await firebaseClient
526         .Child("HomePoints")
527         .Child(auth.GetUid())
```

```
528         .PutAsync(new HomePoints() { username = username, points = points2,  
529         numberOfLogs = 1, toiletFlushCount = 1 });  
530     }  
531 }  
532 }
```

```
1  /*! \class The OutdoorsPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the OutdoorsPointsUpdate ViewModel Class. It updates the data for
   the Outdoors Category of the application. The functions in this class
7  * work by reading in all the chosen data and updating the selected fields and then
   sending this data to back firebase.
8  *
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class OutdoorsPointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int campingCount2 = 0;
23         int picnicCount2 = 0;
24         int plantBushCount2 = 0;
25         int plantFlowerCount2 = 0;
26         int plantTreeCount2 = 0;
27         int scoopCount2 = 0;
28         int fruitGardenCount2 = 0;
29         int herbGardenCount2 = 0;
30         int vegetableGardenCount2 = 0;
31         int birdFeederCount2 = 0;
32
33         string username = "";
34
35         IAuth auth;
36         /** This function updates the points in the Outdoors category by ten points. It
   also increments the number of logs logged in the Outdoors
37         * category by one and increments the number of times this particular action was
   logged by one and sends this data to Firebase.
38         */
39         public async void CampingPoints()
40         {
41
42             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtdb.firebaseio.com/");
43             auth = DependencyService.Get<IAuth>();
44
45             try
46             {
47                 username = (await firebaseClient
48                     .Child("users")
49                     .Child(auth.GetUid())
50                     .OnceSingleAsync<Users>()).username;
51
52                 points2 = (await firebaseClient
53                     .Child("OutdoorsPoints")
54                     .Child(auth.GetUid())
55                     .OnceSingleAsync<OutdoorsPoints>()).points;
56
```

```
57     points2 = points2 + AppConstants.sixPoints;
58
59     numberOfLogs2 = (await firebaseClient
60     .Child("OutdoorsPoints")
61     .Child(auth.GetUid())
62     .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
63
64     numberOfLogs2++;
65
66     campingCount2 = (await firebaseClient
67     .Child("OutdoorsPoints")
68     .Child(auth.GetUid())
69     .OnceSingleAsync<OutdoorsPoints>()).campingCount;
70
71     campingCount2++;
72
73     picnicCount2 = (await firebaseClient
74     .Child("OutdoorsPoints")
75     .Child(auth.GetUid())
76     .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
77
78     plantBushCount2 = (await firebaseClient
79     .Child("OutdoorsPoints")
80     .Child(auth.GetUid())
81     .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
82
83     plantTreeCount2 = (await firebaseClient
84     .Child("OutdoorsPoints")
85     .Child(auth.GetUid())
86     .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
87
88     plantTreeCount2 = (await firebaseClient
89     .Child("OutdoorsPoints")
90     .Child(auth.GetUid())
91     .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
92
93     scoopCount2 = (await firebaseClient
94     .Child("OutdoorsPoints")
95     .Child(auth.GetUid())
96     .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
97
98     fruitGardenCount2 = (await firebaseClient
99     .Child("OutdoorsPoints")
100    .Child(auth.GetUid())
101    .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
102
103    herbGardenCount2 = (await firebaseClient
104    .Child("OutdoorsPoints")
105    .Child(auth.GetUid())
106    .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
107
108    vegetableGardenCount2 = (await firebaseClient
109    .Child("OutdoorsPoints")
110    .Child(auth.GetUid())
111    .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
112
113    birdFeederCount2 = (await firebaseClient
114    .Child("OutdoorsPoints")
115    .Child(auth.GetUid())
116    .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
117
```

```

118     await firebaseClient
119     .Child("OutdoorsPoints")
120     .Child(auth.GetUid())
121     .PutAsync(new OutdoorsPoints()
122     {
123         username = username,
124         points = points2,
125         numberOfLogs = numberOfLogs2,
126         campingCount = campingCount2,
127         picnicCount = picnicCount2,
128         plantBushCount = plantBushCount2,
129         plantFlowerCount = plantBushCount2,
130         plantTreeCount = plantTreeCount2,
131         scoopCount = scoopCount2,
132         fruitGardenCount = fruitGardenCount2,
133         herbGardenCount = herbGardenCount2,
134         vegetableGardenCount = vegetableGardenCount2,
135         birdFeederCount = birdFeederCount2,
136
137     });
138 }
139 catch (FirebaseException)
140 {
141     username = (await firebaseClient
142     .Child("users")
143     .Child(auth.GetUid())
144     .OnceSingleAsync<Users>()).username;
145
146     points2 = AppConstants.sixPoints;
147     await firebaseClient
148     .Child("OutdoorsPoints")
149     .Child(auth.GetUid())
150     .PutAsync(new OutdoorsPoints() { username = username, points = points2,
151 numberOfLogs = 1, campingCount = 1 }); ;
152 }
153 catch (NullReferenceException)
154 {
155     username = (await firebaseClient
156     .Child("users")
157     .Child(auth.GetUid())
158     .OnceSingleAsync<Users>()).username;
159
160     points2 = AppConstants.sixPoints;
161     await firebaseClient
162     .Child("OutdoorsPoints")
163     .Child(auth.GetUid())
164     .PutAsync(new OutdoorsPoints() { username = username, points = points2,
165 numberOfLogs = 1, campingCount = 1 });
166 }
167 /** This function updates the points in the Outdoors category by six points. It
168 also increments the number of logs logged in the Outdoors
169 * category by one and increments the number of times this particular action was
170 logged by one and sends this data to Firebase.
171 */
172 public async void PicnicPoints()
173 {
174     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
175 quake-default-rtdb.firebaseio.com/");
176     auth = DependencyService.Get<IAuth>();

```

```
175
176     try
177     {
178         username = (await firebaseClient
179             .Child("users")
180             .Child(auth.GetUid())
181             .OnceSingleAsync<Users>()).username;
182
183         points2 = (await firebaseClient
184             .Child("OutdoorsPoints")
185             .Child(auth.GetUid())
186             .OnceSingleAsync<OutdoorsPoints>()).points;
187
188         points2 = points2 + AppConstants.sixPoints;
189
190         numberOfLogs2 = (await firebaseClient
191             .Child("OutdoorsPoints")
192             .Child(auth.GetUid())
193             .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
194
195         numberOfLogs2++;
196
197         campingCount2 = (await firebaseClient
198             .Child("OutdoorsPoints")
199             .Child(auth.GetUid())
200             .OnceSingleAsync<OutdoorsPoints>()).campingCount;
201
202         picnicCount2 = (await firebaseClient
203             .Child("OutdoorsPoints")
204             .Child(auth.GetUid())
205             .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
206
207         picnicCount2++;
208
209         plantBushCount2 = (await firebaseClient
210             .Child("OutdoorsPoints")
211             .Child(auth.GetUid())
212             .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
213
214         plantFlowerCount2 = (await firebaseClient
215             .Child("OutdoorsPoints")
216             .Child(auth.GetUid())
217             .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
218
219         plantTreeCount2 = (await firebaseClient
220             .Child("OutdoorsPoints")
221             .Child(auth.GetUid())
222             .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
223
224         scoopCount2 = (await firebaseClient
225             .Child("OutdoorsPoints")
226             .Child(auth.GetUid())
227             .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
228
229         fruitGardenCount2 = (await firebaseClient
230             .Child("OutdoorsPoints")
231             .Child(auth.GetUid())
232             .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
233
234         herbGardenCount2 = (await firebaseClient
235             .Child("OutdoorsPoints")
```

```
236         .Child(auth.GetUid())
237         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
238
239         vegetableGardenCount2 = (await firebaseClient
240         .Child("OutdoorsPoints")
241         .Child(auth.GetUid())
242         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
243
244         birdFeederCount2 = (await firebaseClient
245         .Child("OutdoorsPoints")
246         .Child(auth.GetUid())
247         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
248
249         await firebaseClient
250         .Child("OutdoorsPoints")
251         .Child(auth.GetUid())
252         .PutAsync(new OutdoorsPoints()
253         {
254             username = username,
255             points = points2,
256             numberOfLogs = numberOfLogs2,
257             campingCount = campingCount2,
258             picnicCount = picnicCount2,
259             plantBushCount = plantBushCount2,
260             plantFlowerCount = plantBushCount2,
261             plantTreeCount = plantTreeCount2,
262             scoopCount = scoopCount2,
263             fruitGardenCount = fruitGardenCount2,
264             herbGardenCount = herbGardenCount2,
265             vegetableGardenCount = vegetableGardenCount2,
266             birdFeederCount = birdFeederCount2,
267
268         });
269     }
270     catch (FirebaseException)
271     {
272         username = (await firebaseClient
273         .Child("users")
274         .Child(auth.GetUid())
275         .OnceSingleAsync<Users>()).username;
276
277         points2 = AppConstants.sixPoints;
278         await firebaseClient
279         .Child("OutdoorsPoints")
280         .Child(auth.GetUid())
281         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
282 numberOfLogs = 1, picnicCount = 1 }); ;
283     }
284     catch (NullReferenceException)
285     {
286         username = (await firebaseClient
287         .Child("users")
288         .Child(auth.GetUid())
289         .OnceSingleAsync<Users>()).username;
290
291         points2 = AppConstants.sixPoints;
292         await firebaseClient
293         .Child("OutdoorsPoints")
294         .Child(auth.GetUid())
295         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
```

```
numberOfLogs = 1, picnicCount = 1 });
296     }
297 }
298 /** This function updates the points in the Outdoors category by eight points. It
also increments the number of logs logged in the Outdoors
299 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
300 */
301 public async void PlantBushPoints()
302 {
303
304     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
305     auth = DependencyService.Get<IAuth>();
306
307     try
308     {
309         username = (await firebaseClient
310             .Child("users")
311             .Child(auth.GetUid())
312             .OnceSingleAsync<Users>()).username;
313
314         points2 = (await firebaseClient
315             .Child("OutdoorsPoints")
316             .Child(auth.GetUid())
317             .OnceSingleAsync<OutdoorsPoints>()).points;
318
319         points2 = points2 + AppConstants.eightPoints;
320
321         numberOfLogs2 = (await firebaseClient
322             .Child("OutdoorsPoints")
323             .Child(auth.GetUid())
324             .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
325
326         numberOfLogs2++;
327
328         campingCount2 = (await firebaseClient
329             .Child("OutdoorsPoints")
330             .Child(auth.GetUid())
331             .OnceSingleAsync<OutdoorsPoints>()).campingCount;
332
333         picnicCount2 = (await firebaseClient
334             .Child("OutdoorsPoints")
335             .Child(auth.GetUid())
336             .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
337
338         plantBushCount2 = (await firebaseClient
339             .Child("OutdoorsPoints")
340             .Child(auth.GetUid())
341             .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
342
343         plantBushCount2++;
344
345         plantFlowerCount2 = (await firebaseClient
346             .Child("OutdoorsPoints")
347             .Child(auth.GetUid())
348             .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
349
350         plantTreeCount2 = (await firebaseClient
351             .Child("OutdoorsPoints")
352             .Child(auth.GetUid())
353             .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
```



```
354
355     scoopCount2 = (await firebaseClient
356         .Child("OutdoorsPoints")
357         .Child(auth.GetUid())
358         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
359
360     fruitGardenCount2 = (await firebaseClient
361         .Child("OutdoorsPoints")
362         .Child(auth.GetUid())
363         .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
364
365     herbGardenCount2 = (await firebaseClient
366         .Child("OutdoorsPoints")
367         .Child(auth.GetUid())
368         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
369
370     vegetableGardenCount2 = (await firebaseClient
371         .Child("OutdoorsPoints")
372         .Child(auth.GetUid())
373         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
374
375     birdFeederCount2 = (await firebaseClient
376         .Child("OutdoorsPoints")
377         .Child(auth.GetUid())
378         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
379
380     await firebaseClient
381         .Child("OutdoorsPoints")
382         .Child(auth.GetUid())
383         .PutAsync(new OutdoorsPoints()
384             {
385                 username = username,
386                 points = points2,
387                 numberOfLogs = numberOfLogs2,
388                 campingCount = campingCount2,
389                 picnicCount = picnicCount2,
390                 plantBushCount = plantBushCount2,
391                 plantFlowerCount = plantFlowerCount2,
392                 plantTreeCount = plantTreeCount2,
393                 scoopCount = scoopCount2,
394                 fruitGardenCount = fruitGardenCount2,
395                 herbGardenCount = herbGardenCount2,
396                 vegetableGardenCount = vegetableGardenCount2,
397                 birdFeederCount = birdFeederCount2,
398             });
399     });
400 }
401 catch (FirebaseException)
402 {
403     username = (await firebaseClient
404         .Child("users")
405         .Child(auth.GetUid())
406         .OnceSingleAsync<Users>()).username;
407
408     points2 = AppConstants.eightPoints;
409     await firebaseClient
410         .Child("OutdoorsPoints")
411         .Child(auth.GetUid())
412         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
413     numberOfLogs = 1, plantBushCount = 1 }); ;
```

```
414     }
415     catch (NullReferenceException)
416     {
417         username = (await firebaseClient
418             .Child("users")
419             .Child(auth.GetUid())
420             .OnceSingleAsync<Users>()).username;
421
422         points2 = AppConstants.eightPoints;
423         await firebaseClient
424             .Child("OutdoorsPoints")
425             .Child(auth.GetUid())
426             .PutAsync(new OutdoorsPoints() { username = username, points = points2,
427 numberOfLogs = 1, plantBushCount = 1 });
428     }
429     /** This function updates the points in the Outdoors category by ten points. It
430 also increments the number of logs logged in the Outdoors
431 * category by one and increments the number of times this particular action was
432 logged by one and sends this data to Firebase.
433 */
434     public async void PlantTreePoints()
435     {
436         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
437 quake-default-rtdb.firebaseio.com/");
438         auth = DependencyService.Get<IAuth>();
439
440         try
441         {
442             username = (await firebaseClient
443                 .Child("users")
444                 .Child(auth.GetUid())
445                 .OnceSingleAsync<Users>()).username;
446
447             points2 = (await firebaseClient
448                 .Child("OutdoorsPoints")
449                 .Child(auth.GetUid())
450                 .OnceSingleAsync<OutdoorsPoints>()).points;
451
452             points2 = points2 + AppConstants.tenPoints;
453
454             numberOfLogs2 = (await firebaseClient
455                 .Child("OutdoorsPoints")
456                 .Child(auth.GetUid())
457                 .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
458
459             numberOfLogs2++;
460
461             campingCount2 = (await firebaseClient
462                 .Child("OutdoorsPoints")
463                 .Child(auth.GetUid())
464                 .OnceSingleAsync<OutdoorsPoints>()).campingCount;
465
466             picnicCount2 = (await firebaseClient
467                 .Child("OutdoorsPoints")
468                 .Child(auth.GetUid())
469                 .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
470
471             plantBushCount2 = (await firebaseClient
472                 .Child("OutdoorsPoints")
473                 .Child(auth.GetUid())
```

```
472     .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
473
474     plantFlowerCount2 = (await firebaseClient
475     .Child("OutdoorsPoints")
476     .Child(auth.GetUid())
477     .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
478
479     plantTreeCount2 = (await firebaseClient
480     .Child("OutdoorsPoints")
481     .Child(auth.GetUid())
482     .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
483
484     plantTreeCount2++;
485
486     scoopCount2 = (await firebaseClient
487     .Child("OutdoorsPoints")
488     .Child(auth.GetUid())
489     .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
490
491     fruitGardenCount2 = (await firebaseClient
492     .Child("OutdoorsPoints")
493     .Child(auth.GetUid())
494     .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
495
496     herbGardenCount2 = (await firebaseClient
497     .Child("OutdoorsPoints")
498     .Child(auth.GetUid())
499     .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
500
501     vegetableGardenCount2 = (await firebaseClient
502     .Child("OutdoorsPoints")
503     .Child(auth.GetUid())
504     .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
505
506     birdFeederCount2 = (await firebaseClient
507     .Child("OutdoorsPoints")
508     .Child(auth.GetUid())
509     .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
510
511     await firebaseClient
512     .Child("OutdoorsPoints")
513     .Child(auth.GetUid())
514     .PutAsync(new OutdoorsPoints()
515     {
516         username = username,
517         points = points2,
518         numberOfLogs = numberOfLogs2,
519         campingCount = campingCount2,
520         picnicCount = picnicCount2,
521         plantBushCount = plantBushCount2,
522         plantFlowerCount = plantFlowerCount2,
523         plantTreeCount = plantTreeCount2,
524         scoopCount = scoopCount2,
525         fruitGardenCount = fruitGardenCount2,
526         herbGardenCount = herbGardenCount2,
527         vegetableGardenCount = vegetableGardenCount2,
528         birdFeederCount = birdFeederCount2,
529
530     });
531 }
532 catch (FirebaseException)
```

```
533     {
534         username = (await firebaseClient
535             .Child("users")
536             .Child(auth.GetUid())
537             .OnceSingleAsync<Users>()).username;
538
539         points2 = AppConstants.tenPoints;
540         await firebaseClient
541             .Child("OutdoorsPoints")
542             .Child(auth.GetUid())
543             .PutAsync(new OutdoorsPoints() { username = username, points = points2,
544 numberOfLogs = 1, plantTreeCount = 1 }); ;
545     }
546     catch (NullReferenceException)
547     {
548         username = (await firebaseClient
549             .Child("users")
550             .Child(auth.GetUid())
551             .OnceSingleAsync<Users>()).username;
552
553         points2 = AppConstants.tenPoints;
554         await firebaseClient
555             .Child("OutdoorsPoints")
556             .Child(auth.GetUid())
557             .PutAsync(new OutdoorsPoints() { username = username, points = points2,
558 numberOfLogs = 1, plantTreeCount = 1 }); ;
559     }
560     /** This function updates the points in the Outdoors category by eight points. It
561     also increments the number of logs logged in the Outdoors
562     * category by one and increments the number of times this particular action was
563     logged by one and sends this data to Firebase.
564     */
565     public async void PlantFlowerPoints()
566     {
567         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
568 quake-default-rtdb.firebaseio.com/");
569         auth = DependencyService.Get<IAuth>();
570
571         try
572         {
573             username = (await firebaseClient
574                 .Child("users")
575                 .Child(auth.GetUid())
576                 .OnceSingleAsync<Users>()).username;
577
578             points2 = (await firebaseClient
579                 .Child("OutdoorsPoints")
580                 .Child(auth.GetUid())
581                 .OnceSingleAsync<OutdoorsPoints>()).points;
582
583             points2 = points2 + AppConstants.eightPoints;
584
585             numberOfLogs2 = (await firebaseClient
586                 .Child("OutdoorsPoints")
587                 .Child(auth.GetUid())
588                 .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
589
590             numberOfLogs2++;

```

```
590     campingCount2 = (await firebaseClient
591     .Child("OutdoorsPoints")
592     .Child(auth.GetUid())
593     .OnceSingleAsync<OutdoorsPoints>()).campingCount;
594
595     picnicCount2 = (await firebaseClient
596     .Child("OutdoorsPoints")
597     .Child(auth.GetUid())
598     .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
599
600     plantBushCount2 = (await firebaseClient
601     .Child("OutdoorsPoints")
602     .Child(auth.GetUid())
603     .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
604
605     plantFlowerCount2 = (await firebaseClient
606     .Child("OutdoorsPoints")
607     .Child(auth.GetUid())
608     .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
609
610     plantFlowerCount2++;
611
612     plantTreeCount2 = (await firebaseClient
613     .Child("OutdoorsPoints")
614     .Child(auth.GetUid())
615     .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
616
617     scoopCount2 = (await firebaseClient
618     .Child("OutdoorsPoints")
619     .Child(auth.GetUid())
620     .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
621
622     fruitGardenCount2 = (await firebaseClient
623     .Child("OutdoorsPoints")
624     .Child(auth.GetUid())
625     .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
626
627     herbGardenCount2 = (await firebaseClient
628     .Child("OutdoorsPoints")
629     .Child(auth.GetUid())
630     .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
631
632     vegetableGardenCount2 = (await firebaseClient
633     .Child("OutdoorsPoints")
634     .Child(auth.GetUid())
635     .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
636
637     birdFeederCount2 = (await firebaseClient
638     .Child("OutdoorsPoints")
639     .Child(auth.GetUid())
640     .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
641
642     await firebaseClient
643     .Child("OutdoorsPoints")
644     .Child(auth.GetUid())
645     .PutAsync(new OutdoorsPoints()
646     {
647         username = username,
648         points = points2,
649         numberOfLogs = numberOfLogs2,
650         campingCount = campingCount2,
```

```

651         picnicCount = picnicCount2,
652         plantBushCount = plantBushCount2,
653         plantFlowerCount = plantFlowerCount2,
654         plantTreeCount = plantTreeCount2,
655         scoopCount = scoopCount2,
656         fruitGardenCount = fruitGardenCount2,
657         herbGardenCount = herbGardenCount2,
658         vegetableGardenCount = vegetableGardenCount2,
659         birdFeederCount = birdFeederCount2,
660
661     });
662 }
663 catch (FirebaseException)
664 {
665     username = (await firebaseClient
666         .Child("users")
667         .Child(auth.GetUid())
668         .OnceSingleAsync<Users>()).username;
669
670     points2 = AppConstants.eightPoints;
671     await firebaseClient
672         .Child("OutdoorsPoints")
673         .Child(auth.GetUid())
674         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
675 numberOfLogs = 1, plantTreeCount = 1 }); ;
676 }
677 catch (NullReferenceException)
678 {
679     username = (await firebaseClient
680         .Child("users")
681         .Child(auth.GetUid())
682         .OnceSingleAsync<Users>()).username;
683
684     points2 = AppConstants.eightPoints;
685     await firebaseClient
686         .Child("OutdoorsPoints")
687         .Child(auth.GetUid())
688         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
689 numberOfLogs = 1, plantTreeCount = 1 });
690 }
691 /** This function updates the points in the Outdoors category by four points. It
692 also increments the number of logs logged in the Outdoors
693 * category by one and increments the number of times this particular action was
694 logged by one and sends this data to Firebase.
695 */
696 public async void ScoopPoints()
697 {
698     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
699 quake-default-rtdb.firebaseio.com/");
700     auth = DependencyService.Get<IAuth>();
701
702     try
703     {
704         username = (await firebaseClient
705             .Child("users")
706             .Child(auth.GetUid())
707             .OnceSingleAsync<Users>()).username;
708
709         points2 = (await firebaseClient

```

```
708     .Child("OutdoorsPoints")
709     .Child(auth.GetUid())
710     .OnceSingleAsync<OutdoorsPoints>().points;
711
712     points2 = points2 + AppConstants.fourPoints;
713
714     numberOfLogs2 = (await firebaseClient
715     .Child("OutdoorsPoints")
716     .Child(auth.GetUid())
717     .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
718
719     numberOfLogs2++;
720
721     campingCount2 = (await firebaseClient
722     .Child("OutdoorsPoints")
723     .Child(auth.GetUid())
724     .OnceSingleAsync<OutdoorsPoints>()).campingCount;
725
726     picnicCount2 = (await firebaseClient
727     .Child("OutdoorsPoints")
728     .Child(auth.GetUid())
729     .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
730
731     plantBushCount2 = (await firebaseClient
732     .Child("OutdoorsPoints")
733     .Child(auth.GetUid())
734     .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
735
736     plantFlowerCount2 = (await firebaseClient
737     .Child("OutdoorsPoints")
738     .Child(auth.GetUid())
739     .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
740
741     plantTreeCount2 = (await firebaseClient
742     .Child("OutdoorsPoints")
743     .Child(auth.GetUid())
744     .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
745
746     scoopCount2 = (await firebaseClient
747     .Child("OutdoorsPoints")
748     .Child(auth.GetUid())
749     .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
750
751     scoopCount2++;
752
753     fruitGardenCount2 = (await firebaseClient
754     .Child("OutdoorsPoints")
755     .Child(auth.GetUid())
756     .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
757
758     herbGardenCount2 = (await firebaseClient
759     .Child("OutdoorsPoints")
760     .Child(auth.GetUid())
761     .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
762
763     vegetableGardenCount2 = (await firebaseClient
764     .Child("OutdoorsPoints")
765     .Child(auth.GetUid())
766     .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
767
768     birdFeederCount2 = (await firebaseClient
```

```

769         .Child("OutdoorsPoints")
770         .Child(auth.GetUid())
771         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
772
773         await firebaseClient
774         .Child("OutdoorsPoints")
775         .Child(auth.GetUid())
776         .PutAsync(new OutdoorsPoints()
777         {
778             username = username,
779             points = points2,
780             numberOfLogs = numberOfLogs2,
781             campingCount = campingCount2,
782             picnicCount = picnicCount2,
783             plantBushCount = plantBushCount2,
784             plantFlowerCount = plantFlowerCount2,
785             plantTreeCount = plantTreeCount2,
786             scoopCount = scoopCount2,
787             fruitGardenCount = fruitGardenCount2,
788             herbGardenCount = herbGardenCount2,
789             vegetableGardenCount = vegetableGardenCount2,
790             birdFeederCount = birdFeederCount2,
791
792         });
793     }
794     catch (FirebaseException)
795     {
796         username = (await firebaseClient
797         .Child("users")
798         .Child(auth.GetUid())
799         .OnceSingleAsync<Users>()).username;
800
801         points2 = AppConstants.fourPoints;
802         await firebaseClient
803         .Child("OutdoorsPoints")
804         .Child(auth.GetUid())
805         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
806 numberOfLogs = 1, scoopCount = 1 }); ;
807     }
808     catch (NullReferenceException)
809     {
810         username = (await firebaseClient
811         .Child("users")
812         .Child(auth.GetUid())
813         .OnceSingleAsync<Users>()).username;
814
815         points2 = AppConstants.fourPoints;
816         await firebaseClient
817         .Child("OutdoorsPoints")
818         .Child(auth.GetUid())
819         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
820 numberOfLogs = 1, scoopCount = 1 });
821     }
822     /** This function updates the points in the Outdoors category by ten points. It
823     also increments the number of logs logged in the Outdoors
824     * category by one and increments the number of times this particular action was
825     logged by one and sends this data to Firebase.
826     */
827     public async void FruitGardenPoints()
828     {

```



```
827
828     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
829     auth = DependencyService.Get<IAuth>();
830
831     try
832     {
833         username = (await firebaseClient
834             .Child("users")
835             .Child(auth.GetUid())
836             .OnceSingleAsync<Users>()).username;
837
838         points2 = (await firebaseClient
839             .Child("OutdoorsPoints")
840             .Child(auth.GetUid())
841             .OnceSingleAsync<OutdoorsPoints>()).points;
842
843         points2 = points2 + AppConstants.tenPoints;
844
845         numberOfLogs2 = (await firebaseClient
846             .Child("OutdoorsPoints")
847             .Child(auth.GetUid())
848             .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
849
850         numberOfLogs2++;
851
852         campingCount2 = (await firebaseClient
853             .Child("OutdoorsPoints")
854             .Child(auth.GetUid())
855             .OnceSingleAsync<OutdoorsPoints>()).campingCount;
856
857         picnicCount2 = (await firebaseClient
858             .Child("OutdoorsPoints")
859             .Child(auth.GetUid())
860             .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
861
862         plantBushCount2 = (await firebaseClient
863             .Child("OutdoorsPoints")
864             .Child(auth.GetUid())
865             .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
866
867         plantFlowerCount2 = (await firebaseClient
868             .Child("OutdoorsPoints")
869             .Child(auth.GetUid())
870             .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
871
872         plantTreeCount2 = (await firebaseClient
873             .Child("OutdoorsPoints")
874             .Child(auth.GetUid())
875             .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
876
877         scoopCount2 = (await firebaseClient
878             .Child("OutdoorsPoints")
879             .Child(auth.GetUid())
880             .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
881
882         fruitGardenCount2 = (await firebaseClient
883             .Child("OutdoorsPoints")
884             .Child(auth.GetUid())
885             .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
886
```

```
887         fruitGardenCount2++;
888
889         herbGardenCount2 = (await firebaseClient
890             .Child("OutdoorsPoints")
891             .Child(auth.GetUid())
892             .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
893
894         vegetableGardenCount2 = (await firebaseClient
895             .Child("OutdoorsPoints")
896             .Child(auth.GetUid())
897             .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
898
899         birdFeederCount2 = (await firebaseClient
900             .Child("OutdoorsPoints")
901             .Child(auth.GetUid())
902             .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
903
904         await firebaseClient
905             .Child("OutdoorsPoints")
906             .Child(auth.GetUid())
907             .PutAsync(new OutdoorsPoints()
908             {
909                 username = username,
910                 points = points2,
911                 numberOfLogs = numberOfLogs2,
912                 campingCount = campingCount2,
913                 picnicCount = picnicCount2,
914                 plantBushCount = plantBushCount2,
915                 plantFlowerCount = plantFlowerCount2,
916                 plantTreeCount = plantTreeCount2,
917                 scoopCount = scoopCount2,
918                 fruitGardenCount = fruitGardenCount2,
919                 herbGardenCount = herbGardenCount2,
920                 vegetableGardenCount = vegetableGardenCount2,
921                 birdFeederCount = birdFeederCount2,
922             });
923     }
924 }
925 catch (FirebaseException)
926 {
927     username = (await firebaseClient
928         .Child("users")
929         .Child(auth.GetUid())
930         .OnceSingleAsync<Users>()).username;
931
932     points2 = AppConstants.tenPoints;
933     await firebaseClient
934         .Child("OutdoorsPoints")
935         .Child(auth.GetUid())
936         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
937 numberOfLogs = 1, fruitGardenCount = 1 }); ;
938 }
939 catch (NullReferenceException)
940 {
941     username = (await firebaseClient
942         .Child("users")
943         .Child(auth.GetUid())
944         .OnceSingleAsync<Users>()).username;
945
946     points2 = AppConstants.tenPoints;
```

```
947         await firebaseClient
948             .Child("OutdoorsPoints")
949             .Child(auth.GetUid())
950             .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberOfLogs = 1, fruitGardenCount = 1 });
951     }
952 }
953 /** This function updates the points in the Outdoors category by ten points. It
also increments the number of logs logged in the Outdoors
954     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
955     */
956     public async void HerbGardenPoints()
957     {
958
959         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
960         auth = DependencyService.Get<IAuth>();
961
962         try
963         {
964             username = (await firebaseClient
965                 .Child("users")
966                 .Child(auth.GetUid())
967                 .OnceSingleAsync<Users>()).username;
968
969             points2 = (await firebaseClient
970                 .Child("OutdoorsPoints")
971                 .Child(auth.GetUid())
972                 .OnceSingleAsync<OutdoorsPoints>()).points;
973
974             points2 = points2 + AppConstants.tenPoints;
975
976             numberOfLogs2 = (await firebaseClient
977                 .Child("OutdoorsPoints")
978                 .Child(auth.GetUid())
979                 .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
980
981             numberOfLogs2++;
982
983             campingCount2 = (await firebaseClient
984                 .Child("OutdoorsPoints")
985                 .Child(auth.GetUid())
986                 .OnceSingleAsync<OutdoorsPoints>()).campingCount;
987
988             picnicCount2 = (await firebaseClient
989                 .Child("OutdoorsPoints")
990                 .Child(auth.GetUid())
991                 .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
992
993             plantBushCount2 = (await firebaseClient
994                 .Child("OutdoorsPoints")
995                 .Child(auth.GetUid())
996                 .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
997
998             plantFlowerCount2 = (await firebaseClient
999                 .Child("OutdoorsPoints")
1000                 .Child(auth.GetUid())
1001                 .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
1002
1003             plantTreeCount2 = (await firebaseClient
1004                 .Child("OutdoorsPoints")
```

```
1005     .Child(auth.GetUid())
1006     .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
1007
1008     scoopCount2 = (await firebaseClient
1009     .Child("OutdoorsPoints")
1010     .Child(auth.GetUid())
1011     .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
1012
1013     fruitGardenCount2 = (await firebaseClient
1014     .Child("OutdoorsPoints")
1015     .Child(auth.GetUid())
1016     .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
1017
1018     herbGardenCount2 = (await firebaseClient
1019     .Child("OutdoorsPoints")
1020     .Child(auth.GetUid())
1021     .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
1022
1023     herbGardenCount2++;
1024
1025     vegetableGardenCount2 = (await firebaseClient
1026     .Child("OutdoorsPoints")
1027     .Child(auth.GetUid())
1028     .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
1029
1030     birdFeederCount2 = (await firebaseClient
1031     .Child("OutdoorsPoints")
1032     .Child(auth.GetUid())
1033     .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
1034
1035     await firebaseClient
1036     .Child("OutdoorsPoints")
1037     .Child(auth.GetUid())
1038     .PutAsync(new OutdoorsPoints()
1039     {
1040         username = username,
1041         points = points2,
1042         numberOfLogs = numberOfLogs2,
1043         campingCount = campingCount2,
1044         picnicCount = picnicCount2,
1045         plantBushCount = plantBushCount2,
1046         plantFlowerCount = plantFlowerCount2,
1047         plantTreeCount = plantTreeCount2,
1048         scoopCount = scoopCount2,
1049         fruitGardenCount = fruitGardenCount2,
1050         herbGardenCount = herbGardenCount2,
1051         vegetableGardenCount = vegetableGardenCount2,
1052         birdFeederCount = birdFeederCount2,
1053     });
1054 }
1055 catch (FirebaseException)
1056 {
1057     username = (await firebaseClient
1058     .Child("users")
1059     .Child(auth.GetUid())
1060     .OnceSingleAsync<Users>()).username;
1061
1062     points2 = AppConstants.tenPoints;
1063     await firebaseClient
1064     .Child("OutdoorsPoints")
```

```
1066         .Child(auth.GetUid())
1067         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberOfLogs = 1, herbGardenCount = 1 }); ;
1068     }
1069     }
1070     catch (NullReferenceException)
1071     {
1072         username = (await firebaseClient
1073         .Child("users")
1074         .Child(auth.GetUid())
1075         .OnceSingleAsync<Users>()).username;
1076
1077         points2 = AppConstants.tenPoints;
1078         await firebaseClient
1079         .Child("OutdoorsPoints")
1080         .Child(auth.GetUid())
1081         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberOfLogs = 1, herbGardenCount = 1 });
1082     }
1083 }
1084 /** This function updates the points in the Outdoors category by ten points. It
also increments the number of logs logged in the Outdoors
1085 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1086 */
1087 public async void VegetableGardenPoints()
1088 {
1089
1090     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1091     auth = DependencyService.Get<IAuth>();
1092
1093     try
1094     {
1095         username = (await firebaseClient
1096         .Child("users")
1097         .Child(auth.GetUid())
1098         .OnceSingleAsync<Users>()).username;
1099
1100         points2 = (await firebaseClient
1101         .Child("OutdoorsPoints")
1102         .Child(auth.GetUid())
1103         .OnceSingleAsync<OutdoorsPoints>()).points;
1104
1105         points2 = points2 + AppConstants.tenPoints;
1106
1107         numberOfLogs2 = (await firebaseClient
1108         .Child("OutdoorsPoints")
1109         .Child(auth.GetUid())
1110         .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
1111
1112         numberOfLogs2++;
1113
1114         campingCount2 = (await firebaseClient
1115         .Child("OutdoorsPoints")
1116         .Child(auth.GetUid())
1117         .OnceSingleAsync<OutdoorsPoints>()).campingCount;
1118
1119         picnicCount2 = (await firebaseClient
1120         .Child("OutdoorsPoints")
1121         .Child(auth.GetUid())
1122         .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
```

```
1123
1124     plantBushCount2 = (await firebaseClient
1125         .Child("OutdoorsPoints")
1126         .Child(auth.GetUid())
1127         .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
1128
1129     plantFlowerCount2 = (await firebaseClient
1130         .Child("OutdoorsPoints")
1131         .Child(auth.GetUid())
1132         .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
1133
1134     plantTreeCount2 = (await firebaseClient
1135         .Child("OutdoorsPoints")
1136         .Child(auth.GetUid())
1137         .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
1138
1139     scoopCount2 = (await firebaseClient
1140         .Child("OutdoorsPoints")
1141         .Child(auth.GetUid())
1142         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
1143
1144     fruitGardenCount2 = (await firebaseClient
1145         .Child("OutdoorsPoints")
1146         .Child(auth.GetUid())
1147         .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
1148
1149     herbGardenCount2 = (await firebaseClient
1150         .Child("OutdoorsPoints")
1151         .Child(auth.GetUid())
1152         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
1153
1154     vegetableGardenCount2 = (await firebaseClient
1155         .Child("OutdoorsPoints")
1156         .Child(auth.GetUid())
1157         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
1158
1159     vegetableGardenCount2++;
1160
1161     birdFeederCount2 = (await firebaseClient
1162         .Child("OutdoorsPoints")
1163         .Child(auth.GetUid())
1164         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
1165
1166     await firebaseClient
1167         .Child("OutdoorsPoints")
1168         .Child(auth.GetUid())
1169         .PutAsync(new OutdoorsPoints()
1170     {
1171         username = username,
1172         points = points2,
1173         numberOfLogs = numberOfLogs2,
1174         campingCount = campingCount2,
1175         picnicCount = picnicCount2,
1176         plantBushCount = plantBushCount2,
1177         plantFlowerCount = plantFlowerCount2,
1178         plantTreeCount = plantTreeCount2,
1179         scoopCount = scoopCount2,
1180         fruitGardenCount = fruitGardenCount2,
1181         herbGardenCount = herbGardenCount2,
1182         vegetableGardenCount = vegetableGardenCount2,
1183         birdFeederCount = birdFeederCount2,
```

```
1184         });
1185     });
1186 }
1187 catch (FirebaseException)
1188 {
1189     username = (await firebaseClient
1190         .Child("users")
1191         .Child(auth.GetUid())
1192         .OnceSingleAsync<Users>()).username;
1193
1194     points2 = AppConstants.tenPoints;
1195     await firebaseClient
1196         .Child("OutdoorsPoints")
1197         .Child(auth.GetUid())
1198         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
1199 numberOfLogs = 1, vegetableGardenCount = 1 }); ;
1200 }
1201 catch (NullReferenceException)
1202 {
1203     username = (await firebaseClient
1204         .Child("users")
1205         .Child(auth.GetUid())
1206         .OnceSingleAsync<Users>()).username;
1207
1208     points2 = AppConstants.tenPoints;
1209     await firebaseClient
1210         .Child("OutdoorsPoints")
1211         .Child(auth.GetUid())
1212         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
1213 numberOfLogs = 1, vegetableGardenCount = 1 }); ;
1214 }
1215 /** This function updates the points in the Outdoors category by ten points. It
1216 also increments the number of logs logged in the Outdoors
1217 * category by one and increments the number of times this particular action was
1218 logged by one and sends this data to Firebase.
1219 */
1220 public async void BirdFeederPoints()
1221 {
1222     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1223 quake-default-rtdb.firebaseio.com/");
1224     auth = DependencyService.Get<IAuth>();
1225
1226     try
1227     {
1228         username = (await firebaseClient
1229             .Child("users")
1230             .Child(auth.GetUid())
1231             .OnceSingleAsync<Users>()).username;
1232
1233         points2 = (await firebaseClient
1234             .Child("OutdoorsPoints")
1235             .Child(auth.GetUid())
1236             .OnceSingleAsync<OutdoorsPoints>()).points;
1237
1238         points2 = points2 + AppConstants.tenPoints;
1239
1240         numberOfLogs2 = (await firebaseClient
1241             .Child("OutdoorsPoints")
1242             .Child(auth.GetUid())
```



```
1241         .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
1242
1243         numberOfLogs2++;
1244
1245         campingCount2 = (await firebaseClient
1246             .Child("OutdoorsPoints")
1247             .Child(auth.GetUid())
1248             .OnceSingleAsync<OutdoorsPoints>()).campingCount;
1249
1250         picnicCount2 = (await firebaseClient
1251             .Child("OutdoorsPoints")
1252             .Child(auth.GetUid())
1253             .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
1254
1255         plantBushCount2 = (await firebaseClient
1256             .Child("OutdoorsPoints")
1257             .Child(auth.GetUid())
1258             .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
1259
1260         plantFlowerCount2 = (await firebaseClient
1261             .Child("OutdoorsPoints")
1262             .Child(auth.GetUid())
1263             .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
1264
1265         plantTreeCount2 = (await firebaseClient
1266             .Child("OutdoorsPoints")
1267             .Child(auth.GetUid())
1268             .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
1269
1270         scoopCount2 = (await firebaseClient
1271             .Child("OutdoorsPoints")
1272             .Child(auth.GetUid())
1273             .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
1274
1275         fruitGardenCount2 = (await firebaseClient
1276             .Child("OutdoorsPoints")
1277             .Child(auth.GetUid())
1278             .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
1279
1280         herbGardenCount2 = (await firebaseClient
1281             .Child("OutdoorsPoints")
1282             .Child(auth.GetUid())
1283             .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
1284
1285         vegetableGardenCount2 = (await firebaseClient
1286             .Child("OutdoorsPoints")
1287             .Child(auth.GetUid())
1288             .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
1289
1290         birdFeederCount2 = (await firebaseClient
1291             .Child("OutdoorsPoints")
1292             .Child(auth.GetUid())
1293             .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
1294
1295         birdFeederCount2++;
1296
1297         await firebaseClient
1298             .Child("OutdoorsPoints")
1299             .Child(auth.GetUid())
1300             .PutAsync(new OutdoorsPoints()
1301             {
```



```
1302         username = username,
1303         points = points2,
1304         numberOfLogs = numberOfLogs2,
1305         campingCount = campingCount2,
1306         picnicCount = picnicCount2,
1307         plantBushCount = plantBushCount2,
1308         plantFlowerCount = plantFlowerCount2,
1309         plantTreeCount = plantTreeCount2,
1310         scoopCount = scoopCount2,
1311         fruitGardenCount = fruitGardenCount2,
1312         herbGardenCount = herbGardenCount2,
1313         vegetableGardenCount = vegetableGardenCount2,
1314         birdFeederCount = birdFeederCount2,
1315
1316     });
1317 }
1318 catch (FirebaseException)
1319 {
1320     username = (await firebaseClient
1321         .Child("users")
1322         .Child(auth.GetUid())
1323         .OnceSingleAsync<Users>()).username;
1324
1325     points2 = AppConstants.tenPoints;
1326     await firebaseClient
1327         .Child("OutdoorsPoints")
1328         .Child(auth.GetUid())
1329         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
1330 numberOfLogs = 1, birdFeederCount = 1 }); ;
1331 }
1332 catch (NullReferenceException)
1333 {
1334     username = (await firebaseClient
1335         .Child("users")
1336         .Child(auth.GetUid())
1337         .OnceSingleAsync<Users>()).username;
1338
1339     points2 = AppConstants.tenPoints;
1340     await firebaseClient
1341         .Child("OutdoorsPoints")
1342         .Child(auth.GetUid())
1343         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
1344 numberOfLogs = 1, birdFeederCount = 1 });
1345     }
1346 }
1347 }
```

```
1  /*! \class The PointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the PointsUpdate ViewModel Class. It updates the data for the
7  * Overall Points of user for the application. The functions in this class
8  * work by reading in all the chosen data and updating the selected fields and then sending
9  * this data to back firebase.
10 *
11 */
12 using Application_Green_Quake.Models;
13 using Firebase.Database;
14 using Firebase.Database.Query;
15 using System;
16 using Xamarin.Forms;
17
18 namespace Application_Green_Quake.ViewModels
19 {
20     class PointsUpdate
21     {
22         int points2 = 0;
23         string username = "";
24         string theDate = "";
25         long theTime = 0;
26         int theCount = 0;
27         string currentDate = "";
28         long currentTime = 0;
29
30         IAuth auth;
31         /** This function increases the points of a user by ten points. It also stores the
32         current date and time under the Security Checks node and also
33         * increments the security counter if the point update is on the same day and if it
34         is not then it sets it to zero.
35         */
36         public async void UpdateByTenPoints()
37         {
38             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
39             quake-default-rtdb.firebaseio.com/");
40             auth = DependencyService.Get<IAuth>();
41
42             currentDate = DateTime.UtcNow.ToString("d");
43             currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
44             try
45             {
46                 theDate = (await firebaseClient
47                 .Child("SecurityChecks")
48                 .Child(auth.GetUid())
49                 .OnceSingleAsync<SecurityChecks>()).date;
50
51                 theTime = (await firebaseClient
52                 .Child("SecurityChecks")
53                 .Child(auth.GetUid())
54                 .OnceSingleAsync<SecurityChecks>()).time;
55
56                 theCount = (await firebaseClient
57                 .Child("SecurityChecks")
58                 .Child(auth.GetUid())
59                 .OnceSingleAsync<SecurityChecks>()).counter;
```

```
57
58     if (theDate == currentDate)
59     {
60         theCount++;
61     }
62     else
63     {
64         theCount = 0;
65     }
66
67     await firebaseClient
68     .Child("SecurityChecks")
69     .Child(auth.GetUid())
70     .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = theCount });
71
72     }
73     catch (Exception)
74     {
75         await firebaseClient
76         .Child("SecurityChecks")
77         .Child(auth.GetUid())
78         .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime ,
counter = 1});
79     }
80
81     try
82     {
83         username = (await firebaseClient
84         .Child("users")
85         .Child(auth.GetUid())
86         .OnceSingleAsync<Users>()).username;
87
88         points2 = (await firebaseClient
89         .Child("Points")
90         .Child(auth.GetUid())
91         .OnceSingleAsync<Points>()).points;
92
93         points2 = points2 + AppConstants.tenPoints;
94
95         await firebaseClient
96         .Child("Points")
97         .Child(auth.GetUid())
98         .PutAsync(new Points() { username = username, points = points2 });
99     }
100    catch (FirebaseException)
101    {
102        username = (await firebaseClient
103        .Child("users")
104        .Child(auth.GetUid())
105        .OnceSingleAsync<Users>()).username;
106
107        points2 = AppConstants.tenPoints;
108        await firebaseClient
109        .Child("Points")
110        .Child(auth.GetUid())
111        .PutAsync(new Points() { username = username, points = points2 });
112
113    }
114    catch (NullReferenceException)
115    {
116        username = (await firebaseClient
```

```

117         .Child("users")
118         .Child(auth.GetUid())
119         .OnceSingleAsync<Users>()).username;
120
121         points2 = AppConstants.tenPoints;
122         await firebaseClient
123         .Child("Points")
124         .Child(auth.GetUid())
125         .PutAsync(new Points() { username = username, points = points2 });
126     }
127 }
128 /** 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
129 * increments the security counter if the point update is on the same day and if it
is not then it sets it to zero.
130 */
131 public async void UpdateByEightPoints()
132 {
133     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
134     auth = DependencyService.Get<IAuth>();
135
136     currentDate = DateTime.UtcNow.ToString("d");
137     currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
138     try
139     {
140         theDate = (await firebaseClient
141         .Child("SecurityChecks")
142         .Child(auth.GetUid())
143         .OnceSingleAsync<SecurityChecks>()).date;
144
145         theTime = (await firebaseClient
146         .Child("SecurityChecks")
147         .Child(auth.GetUid())
148         .OnceSingleAsync<SecurityChecks>()).time;
149
150         theCount = (await firebaseClient
151         .Child("SecurityChecks")
152         .Child(auth.GetUid())
153         .OnceSingleAsync<SecurityChecks>()).counter;
154
155         if (theDate == currentDate)
156         {
157             theCount++;
158         }
159         else
160         {
161             theCount = 0;
162         }
163
164         await firebaseClient
165         .Child("SecurityChecks")
166         .Child(auth.GetUid())
167         .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = theCount });
168     }
169     catch (Exception)
170     {
171     }
172     await firebaseClient
173     .Child("SecurityChecks")
174     .Child(auth.GetUid())

```

```

175         .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = 1 });
176     }
177
178     try
179     {
180         username = (await firebaseClient
181             .Child("users")
182             .Child(auth.GetUid())
183             .OnceSingleAsync<Users>()).username;
184
185         points2 = (await firebaseClient
186             .Child("Points")
187             .Child(auth.GetUid())
188             .OnceSingleAsync<Points>()).points;
189
190         points2 = points2 + AppConstants.eightPoints;
191
192         await firebaseClient
193             .Child("Points")
194             .Child(auth.GetUid())
195             .PutAsync(new Points() { username = username, points = points2 });
196     }
197     catch (FirebaseException)
198     {
199         username = (await firebaseClient
200             .Child("users")
201             .Child(auth.GetUid())
202             .OnceSingleAsync<Users>()).username;
203
204         points2 = AppConstants.eightPoints;
205         await firebaseClient
206             .Child("Points")
207             .Child(auth.GetUid())
208             .PutAsync(new Points() { username = username, points = points2 });
209
210     }
211     catch (NullReferenceException)
212     {
213         username = (await firebaseClient
214             .Child("users")
215             .Child(auth.GetUid())
216             .OnceSingleAsync<Users>()).username;
217
218         points2 = AppConstants.eightPoints;
219         await firebaseClient
220             .Child("Points")
221             .Child(auth.GetUid())
222             .PutAsync(new Points() { username = username, points = points2 });
223     }
224 }
225 /** 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
226 * increments the security counter if the point update is on the same day and if it
is not then it sets it to zero.
227 */
228 public async void UpdateBySixPoints()
229 {
230
231     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
232     auth = DependencyService.Get<IAuth>();

```

```
233
234     currentDate = DateTime.UtcNow.ToString("d");
235     currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
236     try
237     {
238         theDate = (await firebaseClient
239             .Child("SecurityChecks")
240             .Child(auth.GetUid())
241             .OnceSingleAsync<SecurityChecks>()).date;
242
243         theTime = (await firebaseClient
244             .Child("SecurityChecks")
245             .Child(auth.GetUid())
246             .OnceSingleAsync<SecurityChecks>()).time;
247
248         theCount = (await firebaseClient
249             .Child("SecurityChecks")
250             .Child(auth.GetUid())
251             .OnceSingleAsync<SecurityChecks>()).counter;
252
253         if (theDate == currentDate)
254         {
255             theCount++;
256         }
257         else
258         {
259             theCount = 0;
260         }
261
262         await firebaseClient
263             .Child("SecurityChecks")
264             .Child(auth.GetUid())
265             .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = theCount });
266
267     }
268     catch (Exception)
269     {
270         await firebaseClient
271             .Child("SecurityChecks")
272             .Child(auth.GetUid())
273             .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = 1 });
274     }
275
276     try
277     {
278         username = (await firebaseClient
279             .Child("users")
280             .Child(auth.GetUid())
281             .OnceSingleAsync<Users>()).username;
282
283         points2 = (await firebaseClient
284             .Child("Points")
285             .Child(auth.GetUid())
286             .OnceSingleAsync<Points>()).points;
287
288         points2 = points2 + AppConstants.sixPoints;
289
290         await firebaseClient
291             .Child("Points")
292             .Child(auth.GetUid())
```

```

293         .PutAsync(new Points() { username = username, points = points2 });
294     }
295     catch (FirebaseException)
296     {
297         username = (await firebaseClient
298             .Child("users")
299             .Child(auth.GetUid())
300             .OnceSingleAsync<Users>()).username;
301
302         points2 = AppConstants.sixPoints;
303         await firebaseClient
304             .Child("Points")
305             .Child(auth.GetUid())
306             .PutAsync(new Points() { username = username, points = points2 });
307
308     }
309     catch (NullReferenceException)
310     {
311         username = (await firebaseClient
312             .Child("users")
313             .Child(auth.GetUid())
314             .OnceSingleAsync<Users>()).username;
315
316         points2 = AppConstants.sixPoints;
317         await firebaseClient
318             .Child("Points")
319             .Child(auth.GetUid())
320             .PutAsync(new Points() { username = username, points = points2 });
321     }
322 }
323 /** This function increases the points of a user by four points. It also stores the
324 current date and time under the Security Checks node and also
325 * increments the security counter if the point update is on the same day and if it
326 is not then it sets it to zero.
327 */
328 public async void UpdateByFourPoints()
329 {
330     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
331 quake-default-rtdb.firebaseio.com/");
332     auth = DependencyService.Get<IAuth>();
333
334     currentDate = DateTime.UtcNow.ToString("d");
335     currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
336     try
337     {
338         theDate = (await firebaseClient
339             .Child("SecurityChecks")
340             .Child(auth.GetUid())
341             .OnceSingleAsync<SecurityChecks>()).date;
342
343         theTime = (await firebaseClient
344             .Child("SecurityChecks")
345             .Child(auth.GetUid())
346             .OnceSingleAsync<SecurityChecks>()).time;
347
348         theCount = (await firebaseClient
349             .Child("SecurityChecks")
350             .Child(auth.GetUid())
351             .OnceSingleAsync<SecurityChecks>()).counter;
352
353         if (theDate == currentDate)

```

```
352     {
353         theCount++;
354     }
355     else
356     {
357         theCount = 0;
358     }
359
360     await firebaseClient
361         .Child("SecurityChecks")
362         .Child(auth.GetUid())
363         .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = theCount });
364     }
365     catch (Exception)
366     {
367         await firebaseClient
368             .Child("SecurityChecks")
369             .Child(auth.GetUid())
370             .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = 1 });
371     }
372 }
373
374 try
375 {
376     username = (await firebaseClient
377         .Child("users")
378         .Child(auth.GetUid())
379         .OnceSingleAsync<Users>()).username;
380
381     points2 = (await firebaseClient
382         .Child("Points")
383         .Child(auth.GetUid())
384         .OnceSingleAsync<Points>()).points;
385
386     points2 = points2 + AppConstants.fourPoints;
387
388     await firebaseClient
389         .Child("Points")
390         .Child(auth.GetUid())
391         .PutAsync(new Points() { username = username, points = points2 });
392 }
393 catch (FirebaseException)
394 {
395     username = (await firebaseClient
396         .Child("users")
397         .Child(auth.GetUid())
398         .OnceSingleAsync<Users>()).username;
399
400     points2 = AppConstants.fourPoints;
401     await firebaseClient
402         .Child("Points")
403         .Child(auth.GetUid())
404         .PutAsync(new Points() { username = username, points = points2 });
405
406 }
407 catch (NullReferenceException)
408 {
409     username = (await firebaseClient
410         .Child("users")
```



```
411         .Child(auth.GetUid())
412         .OnceSingleAsync<Users>()).username;
413
414         points2 = AppConstants.fourPoints;
415         await firebaseClient
416         .Child("Points")
417         .Child(auth.GetUid())
418         .PutAsync(new Points() { username = username, points = points2 });
419     }
420 }
421 /** 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
422 * increments the security counter if the point update is on the same day and if it
is not then it sets it to zero.
423 */
424 public async void UpdateByTwoPoints()
425 {
426     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
427     auth = DependencyService.Get<IAuth>();
428
429     currentDate = DateTime.UtcNow.ToString("d");
430     currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
431     try
432     {
433         theDate = (await firebaseClient
434         .Child("SecurityChecks")
435         .Child(auth.GetUid())
436         .OnceSingleAsync<SecurityChecks>()).date;
437
438         theTime = (await firebaseClient
439         .Child("SecurityChecks")
440         .Child(auth.GetUid())
441         .OnceSingleAsync<SecurityChecks>()).time;
442
443         theCount = (await firebaseClient
444         .Child("SecurityChecks")
445         .Child(auth.GetUid())
446         .OnceSingleAsync<SecurityChecks>()).counter;
447
448         if (theDate == currentDate)
449         {
450             theCount++;
451         }
452         else
453         {
454             theCount = 0;
455         }
456
457         await firebaseClient
458         .Child("SecurityChecks")
459         .Child(auth.GetUid())
460         .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
counter = theCount });
461     }
462     catch (Exception)
463     {
464         await firebaseClient
465         .Child("SecurityChecks")
466         .Child(auth.GetUid())
467         .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
```

```
counter = 1 });
469     }
470
471     try
472     {
473         username = (await firebaseClient
474             .Child("users")
475             .Child(auth.GetUid())
476             .OnceSingleAsync<Users>()).username;
477
478         points2 = (await firebaseClient
479             .Child("Points")
480             .Child(auth.GetUid())
481             .OnceSingleAsync<Points>()).points;
482
483         points2 = points2 + AppConstants.twoPoints;
484
485         await firebaseClient
486             .Child("Points")
487             .Child(auth.GetUid())
488             .PutAsync(new Points() { username = username, points = points2 });
489     }
490     catch (FirebaseException)
491     {
492         username = (await firebaseClient
493             .Child("users")
494             .Child(auth.GetUid())
495             .OnceSingleAsync<Users>()).username;
496
497         points2 = AppConstants.twoPoints;
498         await firebaseClient
499             .Child("Points")
500             .Child(auth.GetUid())
501             .PutAsync(new Points() { username = username, points = points2 });
502     }
503     catch (NullReferenceException)
504     {
505         username = (await firebaseClient
506             .Child("users")
507             .Child(auth.GetUid())
508             .OnceSingleAsync<Users>()).username;
509
510         points2 = AppConstants.twoPoints;
511         await firebaseClient
512             .Child("Points")
513             .Child(auth.GetUid())
514             .PutAsync(new Points() { username = username, points = points2 });
515     }
516 }
517 }
518 }
519 }
```

```

1  /!* \class The SecurityMethods ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SecurityMethods ViewModel Class. It performs security checks
   for the application whenever a log is made in the application. It
7  * prevents the user from logging more than 15 actions per day and more than 1 action per
   60 seconds.
8  *
9  */
10 using System;
11 using System.Threading.Tasks;
12 using Application_Green_Quake.Models;
13 using Firebase.Database;
14 using Firebase.Database.Query;
15 using Xamarin.Forms;
16
17 namespace Application_Green_Quake.ViewModels
18 {
19     class SecurityMethods
20     {
21         IAuth auth;
22         string theDate = "";
23         long theTime = 0;
24         int theCount = 0;
25         string currentDate = "";
26         long currentTime = 0;
27         private long timeDifference = 0;
28
29         /**
30          * 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
31          * the count is 15 and the date is the same as todays date the function returns
   true. Otherwise False.
32          * @return value return true/false
33          */
34         public async Task<bool> DayLimitLock()
35         {
36             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtdb.firebaseio.com/");
37             auth = DependencyService.Get<IAuth>();
38
39             currentDate = DateTime.UtcNow.ToString("d");
40             currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
41
42             try
43             {
44                 theDate = (await firebaseClient
45                     .Child("SecurityChecks")
46                     .Child(auth.GetUid())
47                     .OnceSingleAsync<SecurityChecks>()).date;
48
49                 theCount = (await firebaseClient
50                     .Child("SecurityChecks")
51                     .Child(auth.GetUid())
52                     .OnceSingleAsync<SecurityChecks>()).counter;
53
54                 //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.
55                 if (theCount == 15 && theDate == currentDate)

```

```
56     {
57         return true;
58     }
59
60     return false;
61
62     }
63     catch (Exception)
64     {
65         return false;
66     }
67 }
68 /**
69  * This function gets the time from the SecurityChecks Node in the database and
compares the stored time to the current time. The time difference
70  * 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
71  * function returns true otherwise it returns false.
72  * @return value return true/false
73  */
74 public async Task<bool> TimeLimitLock()
75 {
76     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
77     auth = DependencyService.Get<IAuth>();
78
79     currentDate = DateTime.UtcNow.ToString("d");
80     currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
81
82     try
83     {
84         //Get the time stored in the database.
85         theTime = (await firebaseClient
86             .Child("SecurityChecks")
87             .Child(auth.GetUid())
88             .OnceSingleAsync<SecurityChecks>()).time;
89
90         timeDifference = currentTime - theTime;
91
92         if (timeDifference < 60000)
93         {
94             return true;
95         }
96
97         return false;
98     }
99     catch (Exception)
100     {
101         return false;
102     }
103 }
104 }
105 }
106 }
```

```

1  /*! \class The ShoppingPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ShoppingPointsUpdate ViewModel Class. It updates the data for
   the Shopping Category of the application. The functions in this class
7  * work by reading in all the chosen data and updating the selected fields and then
   sending this data to back firebase.
8  *
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class ShoppingPointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int clothNapkinCount2 = 0;
23         int clothTowelCount2 = 0;
24         int applianceCount2 = 0;
25         int productCount2 = 0;
26         int toothbrushCount2 = 0;
27         int clothesCount2 = 0;
28         int foodCount2 = 0;
29         int localCount2 = 0;
30         int looseLeafCount2 = 0;
31         int organicFoodCount2 = 0;
32         int reusableCount2 = 0;
33         int reBatCount2 = 0;
34         int reBagCount2 = 0;
35
36         string username = "";
37
38         IAuth auth;
39         /** This function updates the points in the Shopping category by two points. It
   also increments the number of logs logged in the Shopping
40         * category by one and increments the number of times this particular action was
   logged by one and sends this data to Firebase.
41         */
42         public async void ClothNapkinsPoints()
43         {
44
45             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtadb.firebaseio.com/");
46             auth = DependencyService.Get<IAuth>();
47
48             try
49             {
50                 username = (await firebaseClient
51                     .Child("users")
52                     .Child(auth.GetUid())
53                     .OnceSingleAsync<Users>()).username;
54
55                 points2 = (await firebaseClient
56                     .Child("ShoppingPoints")

```

```
57     .Child(auth.GetUid())
58     .OnceSingleAsync<ShoppingPoints>()).points;
59
60     points2 = points2 + AppConstants.twoPoints;
61
62     numberOfLogs2 = (await firebaseClient
63     .Child("ShoppingPoints")
64     .Child(auth.GetUid())
65     .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
66
67     numberOfLogs2++;
68
69     clothNapkinCount2 = (await firebaseClient
70     .Child("ShoppingPoints")
71     .Child(auth.GetUid())
72     .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
73
74     clothNapkinCount2++;
75
76     clothTowelCount2 = (await firebaseClient
77     .Child("ShoppingPoints")
78     .Child(auth.GetUid())
79     .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
80
81     applianceCount2 = (await firebaseClient
82     .Child("ShoppingPoints")
83     .Child(auth.GetUid())
84     .OnceSingleAsync<ShoppingPoints>()).applianceCount;
85
86     productCount2 = (await firebaseClient
87     .Child("ShoppingPoints")
88     .Child(auth.GetUid())
89     .OnceSingleAsync<ShoppingPoints>()).productCount;
90
91     toothbrushCount2 = (await firebaseClient
92     .Child("ShoppingPoints")
93     .Child(auth.GetUid())
94     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
95
96     clothesCount2 = (await firebaseClient
97     .Child("ShoppingPoints")
98     .Child(auth.GetUid())
99     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
100
101     foodCount2 = (await firebaseClient
102     .Child("ShoppingPoints")
103     .Child(auth.GetUid())
104     .OnceSingleAsync<ShoppingPoints>()).foodCount;
105
106     localCount2 = (await firebaseClient
107     .Child("ShoppingPoints")
108     .Child(auth.GetUid())
109     .OnceSingleAsync<ShoppingPoints>()).localCount;
110
111     looseLeafCount2 = (await firebaseClient
112     .Child("ShoppingPoints")
113     .Child(auth.GetUid())
114     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
115
116     organicFoodCount2 = (await firebaseClient
117     .Child("ShoppingPoints")
```

```

118         .Child(auth.GetUid())
119         .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
120
121         reusableCount2 = (await firebaseClient
122         .Child("ShoppingPoints")
123         .Child(auth.GetUid())
124         .OnceSingleAsync<ShoppingPoints>()).reusableCount;
125
126         reBatCount2 = (await firebaseClient
127         .Child("ShoppingPoints")
128         .Child(auth.GetUid())
129         .OnceSingleAsync<ShoppingPoints>()).reBatCount;
130
131         reBagCount2 = (await firebaseClient
132         .Child("ShoppingPoints")
133         .Child(auth.GetUid())
134         .OnceSingleAsync<ShoppingPoints>()).reBagCount;
135
136         await firebaseClient
137         .Child("ShoppingPoints")
138         .Child(auth.GetUid())
139         .PutAsync(new ShoppingPoints()
140         {
141             username = username,
142             points = points2,
143             numberOfLogs = numberOfLogs2,
144             clothNapkinCount = clothNapkinCount2,
145             clothTowelCount = clothTowelCount2,
146             applianceCount = applianceCount2,
147             productCount = productCount2,
148             toothbrushCount = toothbrushCount2,
149             clothesCount = clothesCount2,
150             foodCount = foodCount2,
151             localCount = localCount2,
152             looseLeafCount = looseLeafCount2,
153             organicFoodCount = organicFoodCount2,
154             reusableCount = reusableCount2,
155             reBatCount = reBatCount2,
156             reBagCount = reBagCount2,
157         });
158     }
159     catch (FirebaseException)
160     {
161         username = (await firebaseClient
162         .Child("users")
163         .Child(auth.GetUid())
164         .OnceSingleAsync<Users>()).username;
165
166         points2 = AppConstants.twoPoints;
167         await firebaseClient
168         .Child("ShoppingPoints")
169         .Child(auth.GetUid())
170         .PutAsync(new ShoppingPoints() { username = username, points = points2,
171         numberOfLogs = 1, clothNapkinCount = 1 }); ;
172     }
173     catch (NullReferenceException)
174     {
175         username = (await firebaseClient
176         .Child("users")
177         .Child(auth.GetUid())

```

```
178         .OnceSingleAsync<Users>()).username;
179
180         points2 = AppConstants.twoPoints;
181         await firebaseClient
182             .Child("ShoppingPoints")
183             .Child(auth.GetUid())
184             .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, clothNapkinCount = 1 });
185     }
186 }
187 /** This function updates the points in the Shopping category by two points. It
also increments the number of logs logged in the Shopping
188 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
189 */
190 public async void ClothTowelsPoints()
191 {
192
193     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
194     auth = DependencyService.Get<IAuth>();
195
196     try
197     {
198         username = (await firebaseClient
199             .Child("users")
200             .Child(auth.GetUid())
201             .OnceSingleAsync<Users>()).username;
202
203         points2 = (await firebaseClient
204             .Child("ShoppingPoints")
205             .Child(auth.GetUid())
206             .OnceSingleAsync<ShoppingPoints>()).points;
207
208         points2 = points2 + AppConstants.twoPoints;
209
210         numberOfLogs2 = (await firebaseClient
211             .Child("ShoppingPoints")
212             .Child(auth.GetUid())
213             .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
214
215         numberOfLogs2++;
216
217         clothNapkinCount2 = (await firebaseClient
218             .Child("ShoppingPoints")
219             .Child(auth.GetUid())
220             .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
221
222         clothTowelCount2 = (await firebaseClient
223             .Child("ShoppingPoints")
224             .Child(auth.GetUid())
225             .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
226
227         clothTowelCount2++;
228
229         applianceCount2 = (await firebaseClient
230             .Child("ShoppingPoints")
231             .Child(auth.GetUid())
232             .OnceSingleAsync<ShoppingPoints>()).applianceCount;
233
234         productCount2 = (await firebaseClient
235             .Child("ShoppingPoints")
```



```
236     .Child(auth.GetUid())
237     .OnceSingleAsync<ShoppingPoints>()).productCount;
238
239     toothbrushCount2 = (await firebaseClient
240     .Child("ShoppingPoints")
241     .Child(auth.GetUid())
242     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
243
244     clothesCount2 = (await firebaseClient
245     .Child("ShoppingPoints")
246     .Child(auth.GetUid())
247     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
248
249     foodCount2 = (await firebaseClient
250     .Child("ShoppingPoints")
251     .Child(auth.GetUid())
252     .OnceSingleAsync<ShoppingPoints>()).foodCount;
253
254     localCount2 = (await firebaseClient
255     .Child("ShoppingPoints")
256     .Child(auth.GetUid())
257     .OnceSingleAsync<ShoppingPoints>()).localCount;
258
259     looseLeafCount2 = (await firebaseClient
260     .Child("ShoppingPoints")
261     .Child(auth.GetUid())
262     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
263
264     organicFoodCount2 = (await firebaseClient
265     .Child("ShoppingPoints")
266     .Child(auth.GetUid())
267     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
268
269     reusableCount2 = (await firebaseClient
270     .Child("ShoppingPoints")
271     .Child(auth.GetUid())
272     .OnceSingleAsync<ShoppingPoints>()).reusableCount;
273
274     reBatCount2 = (await firebaseClient
275     .Child("ShoppingPoints")
276     .Child(auth.GetUid())
277     .OnceSingleAsync<ShoppingPoints>()).reBatCount;
278
279     reBagCount2 = (await firebaseClient
280     .Child("ShoppingPoints")
281     .Child(auth.GetUid())
282     .OnceSingleAsync<ShoppingPoints>()).reBagCount;
283
284     await firebaseClient
285     .Child("ShoppingPoints")
286     .Child(auth.GetUid())
287     .PutAsync(new ShoppingPoints()
288     {
289         username = username,
290         points = points2,
291         numberOfLogs = numberOfLogs2,
292         clothNapkinCount = clothNapkinCount2,
293         clothTowelCount = clothTowelCount2,
294         applianceCount = applianceCount2,
295         productCount = productCount2,
296         toothbrushCount = toothbrushCount2,
```

```

297         clothesCount = clothesCount2,
298         foodCount = foodCount2,
299         localCount = localCount2,
300         looseLeafCount = looseLeafCount2,
301         organicFoodCount = organicFoodCount2,
302         reusableCount = reusableCount2,
303         reBatCount = reBatCount2,
304         reBagCount = reBagCount2,
305     });
306 }
307 catch (FirebaseException)
308 {
309     username = (await firebaseClient
310         .Child("users")
311         .Child(auth.GetUid())
312         .OnceSingleAsync<Users>()).username;
313
314     points2 = AppConstants.twoPoints;
315     await firebaseClient
316         .Child("ShoppingPoints")
317         .Child(auth.GetUid())
318         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, clothTowelCount = 1 }); ;
319 }
320 }
321 catch (NullReferenceException)
322 {
323     username = (await firebaseClient
324         .Child("users")
325         .Child(auth.GetUid())
326         .OnceSingleAsync<Users>()).username;
327
328     points2 = AppConstants.twoPoints;
329     await firebaseClient
330         .Child("ShoppingPoints")
331         .Child(auth.GetUid())
332         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, clothTowelCount = 1 });
333 }
334 }
335 /** This function updates the points in the Shopping category by two points. It
also increments the number of logs logged in the Shopping
336 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
337 */
338 public async void AppliancePoints()
339 {
340
341     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
342     auth = DependencyService.Get<IAuth>();
343
344     try
345     {
346         username = (await firebaseClient
347             .Child("users")
348             .Child(auth.GetUid())
349             .OnceSingleAsync<Users>()).username;
350
351         points2 = (await firebaseClient
352             .Child("ShoppingPoints")
353             .Child(auth.GetUid())

```

```
354         .OnceSingleAsync<ShoppingPoints>()).points;
355
356         points2 = points2 + AppConstants.twoPoints;
357
358         numberOfLogs2 = (await firebaseClient
359             .Child("ShoppingPoints")
360             .Child(auth.GetUid())
361             .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
362
363         numberOfLogs2++;
364
365         clothNapkinCount2 = (await firebaseClient
366             .Child("ShoppingPoints")
367             .Child(auth.GetUid())
368             .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
369
370         clothTowelCount2 = (await firebaseClient
371             .Child("ShoppingPoints")
372             .Child(auth.GetUid())
373             .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
374
375         applianceCount2 = (await firebaseClient
376             .Child("ShoppingPoints")
377             .Child(auth.GetUid())
378             .OnceSingleAsync<ShoppingPoints>()).applianceCount;
379
380         applianceCount2++;
381
382         productCount2 = (await firebaseClient
383             .Child("ShoppingPoints")
384             .Child(auth.GetUid())
385             .OnceSingleAsync<ShoppingPoints>()).productCount;
386
387         toothbrushCount2 = (await firebaseClient
388             .Child("ShoppingPoints")
389             .Child(auth.GetUid())
390             .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
391
392         clothesCount2 = (await firebaseClient
393             .Child("ShoppingPoints")
394             .Child(auth.GetUid())
395             .OnceSingleAsync<ShoppingPoints>()).clothesCount;
396
397         foodCount2 = (await firebaseClient
398             .Child("ShoppingPoints")
399             .Child(auth.GetUid())
400             .OnceSingleAsync<ShoppingPoints>()).foodCount;
401
402         localCount2 = (await firebaseClient
403             .Child("ShoppingPoints")
404             .Child(auth.GetUid())
405             .OnceSingleAsync<ShoppingPoints>()).localCount;
406
407         looseLeafCount2 = (await firebaseClient
408             .Child("ShoppingPoints")
409             .Child(auth.GetUid())
410             .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
411
412         organicFoodCount2 = (await firebaseClient
413             .Child("ShoppingPoints")
414             .Child(auth.GetUid())
```

```
415         .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
416
417         reusableCount2 = (await firebaseClient
418             .Child("ShoppingPoints")
419             .Child(auth.GetUid())
420             .OnceSingleAsync<ShoppingPoints>()).reusableCount;
421
422         reBatCount2 = (await firebaseClient
423             .Child("ShoppingPoints")
424             .Child(auth.GetUid())
425             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
426
427         reBagCount2 = (await firebaseClient
428             .Child("ShoppingPoints")
429             .Child(auth.GetUid())
430             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
431
432         await firebaseClient
433             .Child("ShoppingPoints")
434             .Child(auth.GetUid())
435             .PutAsync(new ShoppingPoints()
436             {
437                 username = username,
438                 points = points2,
439                 numberOfLogs = numberOfLogs2,
440                 clothNapkinCount = clothNapkinCount2,
441                 clothTowelCount = clothTowelCount2,
442                 applianceCount = applianceCount2,
443                 productCount = productCount2,
444                 toothbrushCount = toothbrushCount2,
445                 clothesCount = clothesCount2,
446                 foodCount = foodCount2,
447                 localCount = localCount2,
448                 looseLeafCount = looseLeafCount2,
449                 organicFoodCount = organicFoodCount2,
450                 reusableCount = reusableCount2,
451                 reBatCount = reBatCount2,
452                 reBagCount = reBagCount2,
453             });
454     }
455     catch (FirebaseException)
456     {
457         username = (await firebaseClient
458             .Child("users")
459             .Child(auth.GetUid())
460             .OnceSingleAsync<Users>()).username;
461
462         points2 = AppConstants.fourPoints;
463         await firebaseClient
464             .Child("ShoppingPoints")
465             .Child(auth.GetUid())
466             .PutAsync(new ShoppingPoints() { username = username, points = points2,
467 numberOfLogs = 1, applianceCount = 1 }); ;
468     }
469     catch (NullReferenceException)
470     {
471         username = (await firebaseClient
472             .Child("users")
473             .Child(auth.GetUid())
474             .OnceSingleAsync<Users>()).username;
```

```
475         points2 = AppConstants.fourPoints;
476         await firebaseClient
477             .Child("ShoppingPoints")
478             .Child(auth.GetUid())
479             .PutAsync(new ShoppingPoints() { username = username, points = points2,
480 numberOfLogs = 1, applianceCount = 1 });
481     }
482 }
483 /** This function updates the points in the Shopping category by four points. It
484 also increments the number of logs logged in the Shopping
485 * category by one and increments the number of times this particular action was
486 logged by one and sends this data to Firebase.
487 */
488 public async void ProductPoints()
489 {
490     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
491 quake-default-rtdb.firebaseio.com/");
492     auth = DependencyService.Get<IAuth>();
493
494     try
495     {
496         username = (await firebaseClient
497             .Child("users")
498             .Child(auth.GetUid())
499             .OnceSingleAsync<Users>()).username;
500
501         points2 = (await firebaseClient
502             .Child("ShoppingPoints")
503             .Child(auth.GetUid())
504             .OnceSingleAsync<ShoppingPoints>()).points;
505
506         points2 = points2 + AppConstants.fourPoints;
507
508         numberOfLogs2 = (await firebaseClient
509             .Child("ShoppingPoints")
510             .Child(auth.GetUid())
511             .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
512
513         numberOfLogs2++;
514
515         clothNapkinCount2 = (await firebaseClient
516             .Child("ShoppingPoints")
517             .Child(auth.GetUid())
518             .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
519
520         clothTowelCount2 = (await firebaseClient
521             .Child("ShoppingPoints")
522             .Child(auth.GetUid())
523             .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
524
525         applianceCount2 = (await firebaseClient
526             .Child("ShoppingPoints")
527             .Child(auth.GetUid())
528             .OnceSingleAsync<ShoppingPoints>()).applianceCount;
529
530         productCount2 = (await firebaseClient
531             .Child("ShoppingPoints")
532             .Child(auth.GetUid())
533             .OnceSingleAsync<ShoppingPoints>()).productCount;
```

```
533     productCount2++;
534
535     toothbrushCount2 = (await firebaseClient
536     .Child("ShoppingPoints")
537     .Child(auth.GetUid())
538     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
539
540     clothesCount2 = (await firebaseClient
541     .Child("ShoppingPoints")
542     .Child(auth.GetUid())
543     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
544
545     foodCount2 = (await firebaseClient
546     .Child("ShoppingPoints")
547     .Child(auth.GetUid())
548     .OnceSingleAsync<ShoppingPoints>()).foodCount;
549
550     localCount2 = (await firebaseClient
551     .Child("ShoppingPoints")
552     .Child(auth.GetUid())
553     .OnceSingleAsync<ShoppingPoints>()).localCount;
554
555     looseLeafCount2 = (await firebaseClient
556     .Child("ShoppingPoints")
557     .Child(auth.GetUid())
558     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
559
560     organicFoodCount2 = (await firebaseClient
561     .Child("ShoppingPoints")
562     .Child(auth.GetUid())
563     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
564
565     reusableCount2 = (await firebaseClient
566     .Child("ShoppingPoints")
567     .Child(auth.GetUid())
568     .OnceSingleAsync<ShoppingPoints>()).reusableCount;
569
570     reBatCount2 = (await firebaseClient
571     .Child("ShoppingPoints")
572     .Child(auth.GetUid())
573     .OnceSingleAsync<ShoppingPoints>()).reBatCount;
574
575     reBagCount2 = (await firebaseClient
576     .Child("ShoppingPoints")
577     .Child(auth.GetUid())
578     .OnceSingleAsync<ShoppingPoints>()).reBagCount;
579
580     await firebaseClient
581     .Child("ShoppingPoints")
582     .Child(auth.GetUid())
583     .PutAsync(new ShoppingPoints()
584     {
585         username = username,
586         points = points2,
587         numberOfLogs = numberOfLogs2,
588         clothNapkinCount = clothNapkinCount2,
589         clothTowelCount = clothTowelCount2,
590         applianceCount = applianceCount2,
591         productCount = productCount2,
592         toothbrushCount = toothbrushCount2,
593         clothesCount = clothesCount2,
```

```
594         foodCount = foodCount2,
595         localCount = localCount2,
596         looseLeafCount = looseLeafCount2,
597         organicFoodCount = organicFoodCount2,
598         reusableCount = reusableCount2,
599         reBatCount = reBatCount2,
600         reBagCount = reBagCount2,
601     });
602 }
603 catch (FirebaseException)
604 {
605     username = (await firebaseClient
606         .Child("users")
607         .Child(auth.GetUid())
608         .OnceSingleAsync<Users>()).username;
609
610     points2 = AppConstants.fourPoints;
611     await firebaseClient
612         .Child("ShoppingPoints")
613         .Child(auth.GetUid())
614         .PutAsync(new ShoppingPoints() { username = username, points = points2,
615 numberOfLogs = 1, productCount = 1 }); ;
616 }
617 catch (NullReferenceException)
618 {
619     username = (await firebaseClient
620         .Child("users")
621         .Child(auth.GetUid())
622         .OnceSingleAsync<Users>()).username;
623
624     points2 = AppConstants.fourPoints;
625     await firebaseClient
626         .Child("ShoppingPoints")
627         .Child(auth.GetUid())
628         .PutAsync(new ShoppingPoints() { username = username, points = points2,
629 numberOfLogs = 1, productCount = 1 });
630 }
631 /** This function updates the points in the Shopping category by six points. It
632 also increments the number of logs logged in the Shopping
633 * category by one and increments the number of times this particular action was
634 logged by one and sends this data to Firebase.
635 */
636 public async void EcoToothbrushPoints()
637 {
638     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
639 quake-default-rtdb.firebaseio.com/");
640     auth = DependencyService.Get<IAuth>();
641
642     try
643     {
644         username = (await firebaseClient
645             .Child("users")
646             .Child(auth.GetUid())
647             .OnceSingleAsync<Users>()).username;
648
649         points2 = (await firebaseClient
650             .Child("ShoppingPoints")
651             .Child(auth.GetUid())
652             .OnceSingleAsync<ShoppingPoints>()).points;
```



```
651
652     points2 = points2 + AppConstants.sixPoints;
653
654     numberOfLogs2 = (await firebaseClient
655     .Child("ShoppingPoints")
656     .Child(auth.GetUid())
657     .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
658
659     numberOfLogs2++;
660
661     clothNapkinCount2 = (await firebaseClient
662     .Child("ShoppingPoints")
663     .Child(auth.GetUid())
664     .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
665
666     clothTowelCount2 = (await firebaseClient
667     .Child("ShoppingPoints")
668     .Child(auth.GetUid())
669     .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
670
671     applianceCount2 = (await firebaseClient
672     .Child("ShoppingPoints")
673     .Child(auth.GetUid())
674     .OnceSingleAsync<ShoppingPoints>()).applianceCount;
675
676     productCount2 = (await firebaseClient
677     .Child("ShoppingPoints")
678     .Child(auth.GetUid())
679     .OnceSingleAsync<ShoppingPoints>()).productCount;
680
681     toothbrushCount2 = (await firebaseClient
682     .Child("ShoppingPoints")
683     .Child(auth.GetUid())
684     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
685
686     toothbrushCount2++;
687
688     clothesCount2 = (await firebaseClient
689     .Child("ShoppingPoints")
690     .Child(auth.GetUid())
691     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
692
693     foodCount2 = (await firebaseClient
694     .Child("ShoppingPoints")
695     .Child(auth.GetUid())
696     .OnceSingleAsync<ShoppingPoints>()).foodCount;
697
698     localCount2 = (await firebaseClient
699     .Child("ShoppingPoints")
700     .Child(auth.GetUid())
701     .OnceSingleAsync<ShoppingPoints>()).localCount;
702
703     looseLeafCount2 = (await firebaseClient
704     .Child("ShoppingPoints")
705     .Child(auth.GetUid())
706     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
707
708     organicFoodCount2 = (await firebaseClient
709     .Child("ShoppingPoints")
710     .Child(auth.GetUid())
711     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
```



```
712         reusableCount2 = (await firebaseClient
713             .Child("ShoppingPoints")
714             .Child(auth.GetUid())
715             .OnceSingleAsync<ShoppingPoints>()).reusableCount;
716
717
718         reBatCount2 = (await firebaseClient
719             .Child("ShoppingPoints")
720             .Child(auth.GetUid())
721             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
722
723
724         reBagCount2 = (await firebaseClient
725             .Child("ShoppingPoints")
726             .Child(auth.GetUid())
727             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
728
729         await firebaseClient
730             .Child("ShoppingPoints")
731             .Child(auth.GetUid())
732             .PutAsync(new ShoppingPoints()
733             {
734                 username = username,
735                 points = points2,
736                 numberOfLogs = numberOfLogs2,
737                 clothNapkinCount = clothNapkinCount2,
738                 clothTowelCount = clothTowelCount2,
739                 applianceCount = applianceCount2,
740                 productCount = productCount2,
741                 toothbrushCount = toothbrushCount2,
742                 clothesCount = clothesCount2,
743                 foodCount = foodCount2,
744                 localCount = localCount2,
745                 looseLeafCount = looseLeafCount2,
746                 organicFoodCount = organicFoodCount2,
747                 reusableCount = reusableCount2,
748                 reBatCount = reBatCount2,
749                 reBagCount = reBagCount2,
750             });
751     }
752     catch (FirebaseException)
753     {
754         username = (await firebaseClient
755             .Child("users")
756             .Child(auth.GetUid())
757             .OnceSingleAsync<Users>()).username;
758
759         points2 = AppConstants.sixPoints;
760         await firebaseClient
761             .Child("ShoppingPoints")
762             .Child(auth.GetUid())
763             .PutAsync(new ShoppingPoints() { username = username, points = points2,
764 numberOfLogs = 1, toothbrushCount = 1 }); ;
765     }
766     catch (NullReferenceException)
767     {
768         username = (await firebaseClient
769             .Child("users")
770             .Child(auth.GetUid())
771             .OnceSingleAsync<Users>()).username;
```

```
772         points2 = AppConstants.sixPoints;
773         await firebaseClient
774             .Child("ShoppingPoints")
775             .Child(auth.GetUid())
776             .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, toothbrushCount = 1 });
777     }
778 }
779 /** This function updates the points in the Shopping category by ten points. It
also increments the number of logs logged in the Shopping
780 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
781 */
782 public async void ClothesPoints()
783 {
784
785     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
786     auth = DependencyService.Get<IAuth>();
787
788     try
789     {
790         username = (await firebaseClient
791             .Child("users")
792             .Child(auth.GetUid())
793             .OnceSingleAsync<Users>()).username;
794
795         points2 = (await firebaseClient
796             .Child("ShoppingPoints")
797             .Child(auth.GetUid())
798             .OnceSingleAsync<ShoppingPoints>()).points;
799
800         points2 = points2 + AppConstants.tenPoints;
801
802         numberOfLogs2 = (await firebaseClient
803             .Child("ShoppingPoints")
804             .Child(auth.GetUid())
805             .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
806
807         numberOfLogs2++;
808
809         clothNapkinCount2 = (await firebaseClient
810             .Child("ShoppingPoints")
811             .Child(auth.GetUid())
812             .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
813
814         clothTowelCount2 = (await firebaseClient
815             .Child("ShoppingPoints")
816             .Child(auth.GetUid())
817             .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
818
819         applianceCount2 = (await firebaseClient
820             .Child("ShoppingPoints")
821             .Child(auth.GetUid())
822             .OnceSingleAsync<ShoppingPoints>()).applianceCount;
823
824         productCount2 = (await firebaseClient
825             .Child("ShoppingPoints")
826             .Child(auth.GetUid())
827             .OnceSingleAsync<ShoppingPoints>()).productCount;
828
829         toothbrushCount2 = (await firebaseClient
```

```
830     .Child("ShoppingPoints")
831     .Child(auth.GetUid())
832     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
833
834     clothesCount2 = (await firebaseClient
835     .Child("ShoppingPoints")
836     .Child(auth.GetUid())
837     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
838
839     clothesCount2++;
840
841     foodCount2 = (await firebaseClient
842     .Child("ShoppingPoints")
843     .Child(auth.GetUid())
844     .OnceSingleAsync<ShoppingPoints>()).foodCount;
845
846     localCount2 = (await firebaseClient
847     .Child("ShoppingPoints")
848     .Child(auth.GetUid())
849     .OnceSingleAsync<ShoppingPoints>()).localCount;
850
851     looseLeafCount2 = (await firebaseClient
852     .Child("ShoppingPoints")
853     .Child(auth.GetUid())
854     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
855
856     organicFoodCount2 = (await firebaseClient
857     .Child("ShoppingPoints")
858     .Child(auth.GetUid())
859     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
860
861     reusableCount2 = (await firebaseClient
862     .Child("ShoppingPoints")
863     .Child(auth.GetUid())
864     .OnceSingleAsync<ShoppingPoints>()).reusableCount;
865
866     reBatCount2 = (await firebaseClient
867     .Child("ShoppingPoints")
868     .Child(auth.GetUid())
869     .OnceSingleAsync<ShoppingPoints>()).reBatCount;
870
871     reBagCount2 = (await firebaseClient
872     .Child("ShoppingPoints")
873     .Child(auth.GetUid())
874     .OnceSingleAsync<ShoppingPoints>()).reBagCount;
875
876     await firebaseClient
877     .Child("ShoppingPoints")
878     .Child(auth.GetUid())
879     .PutAsync(new ShoppingPoints()
880     {
881         username = username,
882         points = points2,
883         numberOfLogs = numberOfLogs2,
884         clothNapkinCount = clothNapkinCount2,
885         clothTowelCount = clothTowelCount2,
886         applianceCount = applianceCount2,
887         productCount = productCount2,
888         toothbrushCount = toothbrushCount2,
889         clothesCount = clothesCount2,
890         foodCount = foodCount2,
```

```

891         localCount = localCount2,
892         looseLeafCount = looseLeafCount2,
893         organicFoodCount = organicFoodCount2,
894         reusableCount = reusableCount2,
895         reBatCount = reBatCount2,
896         reBagCount = reBagCount2,
897     });
898 }
899 catch (FirebaseException)
900 {
901     username = (await firebaseClient
902         .Child("users")
903         .Child(auth.GetUid())
904         .OnceSingleAsync<Users>()).username;
905
906     points2 = AppConstants.tenPoints;
907     await firebaseClient
908         .Child("ShoppingPoints")
909         .Child(auth.GetUid())
910         .PutAsync(new ShoppingPoints() { username = username, points = points2,
911 numberOfLogs = 1, clothesCount = 1 }); ;
912 }
913 catch (NullReferenceException)
914 {
915     username = (await firebaseClient
916         .Child("users")
917         .Child(auth.GetUid())
918         .OnceSingleAsync<Users>()).username;
919
920     points2 = AppConstants.tenPoints;
921     await firebaseClient
922         .Child("ShoppingPoints")
923         .Child(auth.GetUid())
924         .PutAsync(new ShoppingPoints() { username = username, points = points2,
925 numberOfLogs = 1, clothesCount = 1 });
926 }
927 /** This function updates the points in the Shopping category by six points. It
928 also increments the number of logs logged in the Shopping
929 * category by one and increments the number of times this particular action was
930 logged by one and sends this data to Firebase.
931 */
932 public async void FoodInBulkPoints()
933 {
934     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
935 quake-default-rtdb.firebaseio.com/");
936     auth = DependencyService.Get<IAuth>();
937
938     try
939     {
940         username = (await firebaseClient
941             .Child("users")
942             .Child(auth.GetUid())
943             .OnceSingleAsync<Users>()).username;
944
945         points2 = (await firebaseClient
946             .Child("ShoppingPoints")
947             .Child(auth.GetUid())
948             .OnceSingleAsync<ShoppingPoints>()).points;

```

```
948     points2 = points2 + AppConstants.sixPoints;
949
950     numberOfLogs2 = (await firebaseClient
951     .Child("ShoppingPoints")
952     .Child(auth.GetUid())
953     .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
954
955     numberOfLogs2++;
956
957     clothNapkinCount2 = (await firebaseClient
958     .Child("ShoppingPoints")
959     .Child(auth.GetUid())
960     .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
961
962     clothTowelCount2 = (await firebaseClient
963     .Child("ShoppingPoints")
964     .Child(auth.GetUid())
965     .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
966
967     applianceCount2 = (await firebaseClient
968     .Child("ShoppingPoints")
969     .Child(auth.GetUid())
970     .OnceSingleAsync<ShoppingPoints>()).applianceCount;
971
972     productCount2 = (await firebaseClient
973     .Child("ShoppingPoints")
974     .Child(auth.GetUid())
975     .OnceSingleAsync<ShoppingPoints>()).productCount;
976
977     toothbrushCount2 = (await firebaseClient
978     .Child("ShoppingPoints")
979     .Child(auth.GetUid())
980     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
981
982     clothesCount2 = (await firebaseClient
983     .Child("ShoppingPoints")
984     .Child(auth.GetUid())
985     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
986
987     foodCount2 = (await firebaseClient
988     .Child("ShoppingPoints")
989     .Child(auth.GetUid())
990     .OnceSingleAsync<ShoppingPoints>()).foodCount;
991
992     foodCount2++;
993
994     localCount2 = (await firebaseClient
995     .Child("ShoppingPoints")
996     .Child(auth.GetUid())
997     .OnceSingleAsync<ShoppingPoints>()).localCount;
998
999     looseLeafCount2 = (await firebaseClient
1000     .Child("ShoppingPoints")
1001     .Child(auth.GetUid())
1002     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1003
1004     organicFoodCount2 = (await firebaseClient
1005     .Child("ShoppingPoints")
1006     .Child(auth.GetUid())
1007     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1008
```

```
1009     reusableCount2 = (await firebaseClient
1010     .Child("ShoppingPoints")
1011     .Child(auth.GetUid())
1012     .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1013
1014     reBatCount2 = (await firebaseClient
1015     .Child("ShoppingPoints")
1016     .Child(auth.GetUid())
1017     .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1018
1019     reBagCount2 = (await firebaseClient
1020     .Child("ShoppingPoints")
1021     .Child(auth.GetUid())
1022     .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1023
1024     await firebaseClient
1025     .Child("ShoppingPoints")
1026     .Child(auth.GetUid())
1027     .PutAsync(new ShoppingPoints()
1028     {
1029         username = username,
1030         points = points2,
1031         numberOfLogs = numberOfLogs2,
1032         clothNapkinCount = clothNapkinCount2,
1033         clothTowelCount = clothTowelCount2,
1034         applianceCount = applianceCount2,
1035         productCount = productCount2,
1036         toothbrushCount = toothbrushCount2,
1037         clothesCount = clothesCount2,
1038         foodCount = foodCount2,
1039         localCount = localCount2,
1040         looseLeafCount = looseLeafCount2,
1041         organicFoodCount = organicFoodCount2,
1042         reusableCount = reusableCount2,
1043         reBatCount = reBatCount2,
1044         reBagCount = reBagCount2,
1045     });
1046 }
1047 catch (FirebaseException)
1048 {
1049     username = (await firebaseClient
1050     .Child("users")
1051     .Child(auth.GetUid())
1052     .OnceSingleAsync<Users>()).username;
1053
1054     points2 = AppConstants.sixPoints;
1055     await firebaseClient
1056     .Child("ShoppingPoints")
1057     .Child(auth.GetUid())
1058     .PutAsync(new ShoppingPoints() { username = username, points = points2,
1059     numberOfLogs = 1, foodCount = 1 }); ;
1060 }
1061 catch (NullReferenceException)
1062 {
1063     username = (await firebaseClient
1064     .Child("users")
1065     .Child(auth.GetUid())
1066     .OnceSingleAsync<Users>()).username;
1067
1068     points2 = AppConstants.sixPoints;
```

```
1069         await firebaseClient
1070             .Child("ShoppingPoints")
1071             .Child(auth.GetUid())
1072             .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, foodCount = 1 });
1073     }
1074 }
1075 /** This function updates the points in the Shopping category by eight points. It
also increments the number of logs logged in the Shopping
1076     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1077     */
1078     public async void LocalProductPoints()
1079     {
1080
1081         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1082         auth = DependencyService.Get<IAuth>();
1083
1084         try
1085         {
1086             username = (await firebaseClient
1087                 .Child("users")
1088                 .Child(auth.GetUid())
1089                 .OnceSingleAsync<Users>()).username;
1090
1091             points2 = (await firebaseClient
1092                 .Child("ShoppingPoints")
1093                 .Child(auth.GetUid())
1094                 .OnceSingleAsync<ShoppingPoints>()).points;
1095
1096             points2 = points2 + AppConstants.eightPoints;
1097
1098             numberOfLogs2 = (await firebaseClient
1099                 .Child("ShoppingPoints")
1100                 .Child(auth.GetUid())
1101                 .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
1102
1103             numberOfLogs2++;
1104
1105             clothNapkinCount2 = (await firebaseClient
1106                 .Child("ShoppingPoints")
1107                 .Child(auth.GetUid())
1108                 .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1109
1110             clothTowelCount2 = (await firebaseClient
1111                 .Child("ShoppingPoints")
1112                 .Child(auth.GetUid())
1113                 .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1114
1115             applianceCount2 = (await firebaseClient
1116                 .Child("ShoppingPoints")
1117                 .Child(auth.GetUid())
1118                 .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1119
1120             productCount2 = (await firebaseClient
1121                 .Child("ShoppingPoints")
1122                 .Child(auth.GetUid())
1123                 .OnceSingleAsync<ShoppingPoints>()).productCount;
1124
1125             toothbrushCount2 = (await firebaseClient
1126                 .Child("ShoppingPoints")
```



```
1127     .Child(auth.GetUid())
1128     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
1129
1130     clothesCount2 = (await firebaseClient
1131     .Child("ShoppingPoints")
1132     .Child(auth.GetUid())
1133     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
1134
1135     foodCount2 = (await firebaseClient
1136     .Child("ShoppingPoints")
1137     .Child(auth.GetUid())
1138     .OnceSingleAsync<ShoppingPoints>()).foodCount;
1139
1140     localCount2 = (await firebaseClient
1141     .Child("ShoppingPoints")
1142     .Child(auth.GetUid())
1143     .OnceSingleAsync<ShoppingPoints>()).localCount;
1144
1145     localCount2++;
1146
1147     looseLeafCount2 = (await firebaseClient
1148     .Child("ShoppingPoints")
1149     .Child(auth.GetUid())
1150     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1151
1152     organicFoodCount2 = (await firebaseClient
1153     .Child("ShoppingPoints")
1154     .Child(auth.GetUid())
1155     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1156
1157     reusableCount2 = (await firebaseClient
1158     .Child("ShoppingPoints")
1159     .Child(auth.GetUid())
1160     .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1161
1162     reBatCount2 = (await firebaseClient
1163     .Child("ShoppingPoints")
1164     .Child(auth.GetUid())
1165     .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1166
1167     reBagCount2 = (await firebaseClient
1168     .Child("ShoppingPoints")
1169     .Child(auth.GetUid())
1170     .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1171
1172     await firebaseClient
1173     .Child("ShoppingPoints")
1174     .Child(auth.GetUid())
1175     .PutAsync(new ShoppingPoints()
1176     {
1177         username = username,
1178         points = points2,
1179         numberOfLogs = numberOfLogs2,
1180         clothNapkinCount = clothNapkinCount2,
1181         clothTowelCount = clothTowelCount2,
1182         applianceCount = applianceCount2,
1183         productCount = productCount2,
1184         toothbrushCount = toothbrushCount2,
1185         clothesCount = clothesCount2,
1186         foodCount = foodCount2,
1187         localCount = localCount2,
```



```
1188         looseLeafCount = looseLeafCount2,
1189         organicFoodCount = organicFoodCount2,
1190         reusableCount = reusableCount2,
1191         reBatCount = reBatCount2,
1192         reBagCount = reBagCount2,
1193     });
1194 }
1195 catch (FirebaseException)
1196 {
1197     username = (await firebaseClient
1198         .Child("users")
1199         .Child(auth.GetUid())
1200         .OnceSingleAsync<Users>()).username;
1201
1202     points2 = AppConstants.eightPoints;
1203     await firebaseClient
1204         .Child("ShoppingPoints")
1205         .Child(auth.GetUid())
1206         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, productCount = 1 }); ;
1207
1208 }
1209 catch (NullReferenceException)
1210 {
1211     username = (await firebaseClient
1212         .Child("users")
1213         .Child(auth.GetUid())
1214         .OnceSingleAsync<Users>()).username;
1215
1216     points2 = AppConstants.eightPoints;
1217     await firebaseClient
1218         .Child("ShoppingPoints")
1219         .Child(auth.GetUid())
1220         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, productCount = 1 }); ;
1221     }
1222 }
1223 /** This function updates the points in the Shopping category by four points. It
also increments the number of logs logged in the Shopping
1224 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1225 */
1226 public async void TeaPoints()
1227 {
1228
1229     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1230     auth = DependencyService.Get<IAuth>();
1231
1232     try
1233     {
1234         username = (await firebaseClient
1235             .Child("users")
1236             .Child(auth.GetUid())
1237             .OnceSingleAsync<Users>()).username;
1238
1239         points2 = (await firebaseClient
1240             .Child("ShoppingPoints")
1241             .Child(auth.GetUid())
1242             .OnceSingleAsync<ShoppingPoints>()).points;
1243
1244         points2 = points2 + AppConstants.fourPoints;
```

```
1245
1246     numberOfLogs2 = (await firebaseClient
1247     .Child("ShoppingPoints")
1248     .Child(auth.GetUid())
1249     .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
1250
1251     numberOfLogs2++;
1252
1253     clothNapkinCount2 = (await firebaseClient
1254     .Child("ShoppingPoints")
1255     .Child(auth.GetUid())
1256     .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1257
1258     clothTowelCount2 = (await firebaseClient
1259     .Child("ShoppingPoints")
1260     .Child(auth.GetUid())
1261     .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1262
1263     applianceCount2 = (await firebaseClient
1264     .Child("ShoppingPoints")
1265     .Child(auth.GetUid())
1266     .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1267
1268     productCount2 = (await firebaseClient
1269     .Child("ShoppingPoints")
1270     .Child(auth.GetUid())
1271     .OnceSingleAsync<ShoppingPoints>()).productCount;
1272
1273     toothbrushCount2 = (await firebaseClient
1274     .Child("ShoppingPoints")
1275     .Child(auth.GetUid())
1276     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
1277
1278     clothesCount2 = (await firebaseClient
1279     .Child("ShoppingPoints")
1280     .Child(auth.GetUid())
1281     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
1282
1283     foodCount2 = (await firebaseClient
1284     .Child("ShoppingPoints")
1285     .Child(auth.GetUid())
1286     .OnceSingleAsync<ShoppingPoints>()).foodCount;
1287
1288     localCount2 = (await firebaseClient
1289     .Child("ShoppingPoints")
1290     .Child(auth.GetUid())
1291     .OnceSingleAsync<ShoppingPoints>()).localCount;
1292
1293     looseLeafCount2 = (await firebaseClient
1294     .Child("ShoppingPoints")
1295     .Child(auth.GetUid())
1296     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1297
1298     looseLeafCount2++;
1299
1300     organicFoodCount2 = (await firebaseClient
1301     .Child("ShoppingPoints")
1302     .Child(auth.GetUid())
1303     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1304
1305     reusableCount2 = (await firebaseClient
```

```
1306     .Child("ShoppingPoints")
1307     .Child(auth.GetUid())
1308     .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1309
1310     reBatCount2 = (await firebaseClient
1311     .Child("ShoppingPoints")
1312     .Child(auth.GetUid())
1313     .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1314
1315     reBagCount2 = (await firebaseClient
1316     .Child("ShoppingPoints")
1317     .Child(auth.GetUid())
1318     .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1319
1320     await firebaseClient
1321     .Child("ShoppingPoints")
1322     .Child(auth.GetUid())
1323     .PutAsync(new ShoppingPoints()
1324     {
1325         username = username,
1326         points = points2,
1327         numberOfLogs = numberOfLogs2,
1328         clothNapkinCount = clothNapkinCount2,
1329         clothTowelCount = clothTowelCount2,
1330         applianceCount = applianceCount2,
1331         productCount = productCount2,
1332         toothbrushCount = toothbrushCount2,
1333         clothesCount = clothesCount2,
1334         foodCount = foodCount2,
1335         localCount = localCount2,
1336         looseLeafCount = looseLeafCount2,
1337         organicFoodCount = organicFoodCount2,
1338         reusableCount = reusableCount2,
1339         reBatCount = reBatCount2,
1340         reBagCount = reBagCount2,
1341     });
1342 }
1343 catch (FirebaseException)
1344 {
1345     username = (await firebaseClient
1346     .Child("users")
1347     .Child(auth.GetUid())
1348     .OnceSingleAsync<Users>()).username;
1349
1350     points2 = AppConstants.fourPoints;
1351     await firebaseClient
1352     .Child("ShoppingPoints")
1353     .Child(auth.GetUid())
1354     .PutAsync(new ShoppingPoints() { username = username, points = points2,
1355 numberOfLogs = 1, looseLeafCount = 1 }); ;
1356 }
1357 catch (NullReferenceException)
1358 {
1359     username = (await firebaseClient
1360     .Child("users")
1361     .Child(auth.GetUid())
1362     .OnceSingleAsync<Users>()).username;
1363
1364     points2 = AppConstants.fourPoints;
1365     await firebaseClient
```

```
1366         .Child("ShoppingPoints")
1367         .Child(auth.GetUid())
1368         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, looseLeafCount = 1 });
1369     }
1370 }
1371 /** This function updates the points in the Shopping category by eight points. It
also increments the number of logs logged in the Shopping
1372 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1373 */
1374 public async void OrganicPoints()
1375 {
1376
1377     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1378     auth = DependencyService.Get<IAuth>();
1379
1380     try
1381     {
1382         username = (await firebaseClient
1383         .Child("users")
1384         .Child(auth.GetUid())
1385         .OnceSingleAsync<Users>()).username;
1386
1387         points2 = (await firebaseClient
1388         .Child("ShoppingPoints")
1389         .Child(auth.GetUid())
1390         .OnceSingleAsync<ShoppingPoints>()).points;
1391
1392         points2 = points2 + AppConstants.eightPoints;
1393
1394         numberOfLogs2 = (await firebaseClient
1395         .Child("ShoppingPoints")
1396         .Child(auth.GetUid())
1397         .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
1398
1399         numberOfLogs2++;
1400
1401         clothNapkinCount2 = (await firebaseClient
1402         .Child("ShoppingPoints")
1403         .Child(auth.GetUid())
1404         .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1405
1406         clothTowelCount2 = (await firebaseClient
1407         .Child("ShoppingPoints")
1408         .Child(auth.GetUid())
1409         .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1410
1411         applianceCount2 = (await firebaseClient
1412         .Child("ShoppingPoints")
1413         .Child(auth.GetUid())
1414         .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1415
1416         productCount2 = (await firebaseClient
1417         .Child("ShoppingPoints")
1418         .Child(auth.GetUid())
1419         .OnceSingleAsync<ShoppingPoints>()).productCount;
1420
1421         toothbrushCount2 = (await firebaseClient
1422         .Child("ShoppingPoints")
1423         .Child(auth.GetUid())
```

```
1424         .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
1425
1426         clothesCount2 = (await firebaseClient
1427             .Child("ShoppingPoints")
1428             .Child(auth.GetUid())
1429             .OnceSingleAsync<ShoppingPoints>()).clothesCount;
1430
1431         foodCount2 = (await firebaseClient
1432             .Child("ShoppingPoints")
1433             .Child(auth.GetUid())
1434             .OnceSingleAsync<ShoppingPoints>()).foodCount;
1435
1436         localCount2 = (await firebaseClient
1437             .Child("ShoppingPoints")
1438             .Child(auth.GetUid())
1439             .OnceSingleAsync<ShoppingPoints>()).localCount;
1440
1441         looseLeafCount2 = (await firebaseClient
1442             .Child("ShoppingPoints")
1443             .Child(auth.GetUid())
1444             .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1445
1446         organicFoodCount2 = (await firebaseClient
1447             .Child("ShoppingPoints")
1448             .Child(auth.GetUid())
1449             .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1450
1451         organicFoodCount2++;
1452
1453         reusableCount2 = (await firebaseClient
1454             .Child("ShoppingPoints")
1455             .Child(auth.GetUid())
1456             .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1457
1458         reBatCount2 = (await firebaseClient
1459             .Child("ShoppingPoints")
1460             .Child(auth.GetUid())
1461             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1462
1463         reBagCount2 = (await firebaseClient
1464             .Child("ShoppingPoints")
1465             .Child(auth.GetUid())
1466             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1467
1468         await firebaseClient
1469             .Child("ShoppingPoints")
1470             .Child(auth.GetUid())
1471             .PutAsync(new ShoppingPoints()
1472             {
1473                 username = username,
1474                 points = points2,
1475                 numberOfLogs = numberOfLogs2,
1476                 clothNapkinCount = clothNapkinCount2,
1477                 clothTowelCount = clothTowelCount2,
1478                 applianceCount = applianceCount2,
1479                 productCount = productCount2,
1480                 toothbrushCount = toothbrushCount2,
1481                 clothesCount = clothesCount2,
1482                 foodCount = foodCount2,
1483                 localCount = localCount2,
1484                 looseLeafCount = looseLeafCount2,
```

```

1485         organicFoodCount = organicFoodCount2,
1486         reusableCount = reusableCount2,
1487         reBatCount = reBatCount2,
1488         reBagCount = reBagCount2,
1489     });
1490 }
1491 catch (FirebaseException)
1492 {
1493     username = (await firebaseClient
1494         .Child("users")
1495         .Child(auth.GetUid())
1496         .OnceSingleAsync<Users>()).username;
1497
1498     points2 = AppConstants.eightPoints;
1499     await firebaseClient
1500         .Child("ShoppingPoints")
1501         .Child(auth.GetUid())
1502         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, organicFoodCount = 1 }); ;
1503
1504 }
1505 catch (NullReferenceException)
1506 {
1507     username = (await firebaseClient
1508         .Child("users")
1509         .Child(auth.GetUid())
1510         .OnceSingleAsync<Users>()).username;
1511
1512     points2 = AppConstants.eightPoints;
1513     await firebaseClient
1514         .Child("ShoppingPoints")
1515         .Child(auth.GetUid())
1516         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, organicFoodCount = 1 }); ;
1517     }
1518 }
1519 /** This function updates the points in the Shopping category by eight points. It
also increments the number of logs logged in the Shopping
1520 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1521 */
1522 public async void ReWaterPoints()
1523 {
1524
1525     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1526     auth = DependencyService.Get<IAuth>();
1527
1528     try
1529     {
1530         username = (await firebaseClient
1531             .Child("users")
1532             .Child(auth.GetUid())
1533             .OnceSingleAsync<Users>()).username;
1534
1535         points2 = (await firebaseClient
1536             .Child("ShoppingPoints")
1537             .Child(auth.GetUid())
1538             .OnceSingleAsync<ShoppingPoints>()).points;
1539
1540         points2 = points2 + AppConstants.eightPoints;
1541

```

```
1542     numberOfLogs2 = (await firebaseClient
1543     .Child("ShoppingPoints")
1544     .Child(auth.GetUid())
1545     .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
1546
1547     numberOfLogs2++;
1548
1549     clothNapkinCount2 = (await firebaseClient
1550     .Child("ShoppingPoints")
1551     .Child(auth.GetUid())
1552     .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1553
1554     clothTowelCount2 = (await firebaseClient
1555     .Child("ShoppingPoints")
1556     .Child(auth.GetUid())
1557     .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1558
1559     applianceCount2 = (await firebaseClient
1560     .Child("ShoppingPoints")
1561     .Child(auth.GetUid())
1562     .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1563
1564     productCount2 = (await firebaseClient
1565     .Child("ShoppingPoints")
1566     .Child(auth.GetUid())
1567     .OnceSingleAsync<ShoppingPoints>()).productCount;
1568
1569     toothbrushCount2 = (await firebaseClient
1570     .Child("ShoppingPoints")
1571     .Child(auth.GetUid())
1572     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
1573
1574     clothesCount2 = (await firebaseClient
1575     .Child("ShoppingPoints")
1576     .Child(auth.GetUid())
1577     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
1578
1579     foodCount2 = (await firebaseClient
1580     .Child("ShoppingPoints")
1581     .Child(auth.GetUid())
1582     .OnceSingleAsync<ShoppingPoints>()).foodCount;
1583
1584     localCount2 = (await firebaseClient
1585     .Child("ShoppingPoints")
1586     .Child(auth.GetUid())
1587     .OnceSingleAsync<ShoppingPoints>()).localCount;
1588
1589     looseLeafCount2 = (await firebaseClient
1590     .Child("ShoppingPoints")
1591     .Child(auth.GetUid())
1592     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1593
1594     organicFoodCount2 = (await firebaseClient
1595     .Child("ShoppingPoints")
1596     .Child(auth.GetUid())
1597     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1598
1599     reusableCount2 = (await firebaseClient
1600     .Child("ShoppingPoints")
1601     .Child(auth.GetUid())
1602     .OnceSingleAsync<ShoppingPoints>()).reusableCount;
```



```
1603
1604     reusableCount2++;
1605
1606     reBatCount2 = (await firebaseClient
1607     .Child("ShoppingPoints")
1608     .Child(auth.GetUid())
1609     .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1610
1611     reBagCount2 = (await firebaseClient
1612     .Child("ShoppingPoints")
1613     .Child(auth.GetUid())
1614     .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1615
1616     await firebaseClient
1617     .Child("ShoppingPoints")
1618     .Child(auth.GetUid())
1619     .PutAsync(new ShoppingPoints()
1620     {
1621         username = username,
1622         points = points2,
1623         numberOfLogs = numberOfLogs2,
1624         clothNapkinCount = clothNapkinCount2,
1625         clothTowelCount = clothTowelCount2,
1626         applianceCount = applianceCount2,
1627         productCount = productCount2,
1628         toothbrushCount = toothbrushCount2,
1629         clothesCount = clothesCount2,
1630         foodCount = foodCount2,
1631         localCount = localCount2,
1632         looseLeafCount = looseLeafCount2,
1633         organicFoodCount = organicFoodCount2,
1634         reusableCount = reusableCount2,
1635         reBatCount = reBatCount2,
1636         reBagCount = reBagCount2,
1637     });
1638 }
1639 catch (FirebaseException)
1640 {
1641     username = (await firebaseClient
1642     .Child("users")
1643     .Child(auth.GetUid())
1644     .OnceSingleAsync<Users>()).username;
1645
1646     points2 = AppConstants.eightPoints;
1647     await firebaseClient
1648     .Child("ShoppingPoints")
1649     .Child(auth.GetUid())
1650     .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, reusableCount = 1 }); ;
1651 }
1652 catch (NullReferenceException)
1653 {
1654     username = (await firebaseClient
1655     .Child("users")
1656     .Child(auth.GetUid())
1657     .OnceSingleAsync<Users>()).username;
1658
1659     points2 = AppConstants.eightPoints;
1660     await firebaseClient
1661     .Child("ShoppingPoints")
```



```
1663         .Child(auth.GetUid())
1664         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, reusableCount = 1 });
1665     }
1666 }
1667 /** This function updates the points in the Shopping category by six points. It
also increments the number of logs logged in the Shopping
1668 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
1669 */
1670 public async void ReBattereisPoints()
1671 {
1672
1673     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
1674     auth = DependencyService.Get<IAuth>();
1675
1676     try
1677     {
1678         username = (await firebaseClient
1679             .Child("users")
1680             .Child(auth.GetUid())
1681             .OnceSingleAsync<Users>()).username;
1682
1683         points2 = (await firebaseClient
1684             .Child("ShoppingPoints")
1685             .Child(auth.GetUid())
1686             .OnceSingleAsync<ShoppingPoints>()).points;
1687
1688         points2 = points2 + AppConstants.sixPoints;
1689
1690         numberOfLogs2 = (await firebaseClient
1691             .Child("ShoppingPoints")
1692             .Child(auth.GetUid())
1693             .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
1694
1695         numberOfLogs2++;
1696
1697         clothNapkinCount2 = (await firebaseClient
1698             .Child("ShoppingPoints")
1699             .Child(auth.GetUid())
1700             .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1701
1702         clothTowelCount2 = (await firebaseClient
1703             .Child("ShoppingPoints")
1704             .Child(auth.GetUid())
1705             .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1706
1707         applianceCount2 = (await firebaseClient
1708             .Child("ShoppingPoints")
1709             .Child(auth.GetUid())
1710             .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1711
1712         productCount2 = (await firebaseClient
1713             .Child("ShoppingPoints")
1714             .Child(auth.GetUid())
1715             .OnceSingleAsync<ShoppingPoints>()).productCount;
1716
1717         toothbrushCount2 = (await firebaseClient
1718             .Child("ShoppingPoints")
1719             .Child(auth.GetUid())
1720             .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
```

```
1721
1722         clothesCount2 = (await firebaseClient
1723             .Child("ShoppingPoints")
1724             .Child(auth.GetUid())
1725             .OnceSingleAsync<ShoppingPoints>()).clothesCount;
1726
1727         foodCount2 = (await firebaseClient
1728             .Child("ShoppingPoints")
1729             .Child(auth.GetUid())
1730             .OnceSingleAsync<ShoppingPoints>()).foodCount;
1731
1732         localCount2 = (await firebaseClient
1733             .Child("ShoppingPoints")
1734             .Child(auth.GetUid())
1735             .OnceSingleAsync<ShoppingPoints>()).localCount;
1736
1737         looseLeafCount2 = (await firebaseClient
1738             .Child("ShoppingPoints")
1739             .Child(auth.GetUid())
1740             .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1741
1742         organicFoodCount2 = (await firebaseClient
1743             .Child("ShoppingPoints")
1744             .Child(auth.GetUid())
1745             .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1746
1747         reusableCount2 = (await firebaseClient
1748             .Child("ShoppingPoints")
1749             .Child(auth.GetUid())
1750             .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1751
1752         reBatCount2 = (await firebaseClient
1753             .Child("ShoppingPoints")
1754             .Child(auth.GetUid())
1755             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1756
1757         reBatCount2++;
1758
1759         reBagCount2 = (await firebaseClient
1760             .Child("ShoppingPoints")
1761             .Child(auth.GetUid())
1762             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1763
1764         await firebaseClient
1765             .Child("ShoppingPoints")
1766             .Child(auth.GetUid())
1767             .PutAsync(new ShoppingPoints()
1768             {
1769                 username = username,
1770                 points = points2,
1771                 numberOfLogs = numberOfLogs2,
1772                 clothNapkinCount = clothNapkinCount2,
1773                 clothTowelCount = clothTowelCount2,
1774                 applianceCount = applianceCount2,
1775                 productCount = productCount2,
1776                 toothbrushCount = toothbrushCount2,
1777                 clothesCount = clothesCount2,
1778                 foodCount = foodCount2,
1779                 localCount = localCount2,
1780                 looseLeafCount = looseLeafCount2,
1781                 organicFoodCount = organicFoodCount2,
```

```

1782         reusableCount = reusableCount2,
1783         reBatCount = reBatCount2,
1784         reBagCount = reBagCount2,
1785     });
1786 }
1787 catch (FirebaseException)
1788 {
1789     username = (await firebaseClient
1790         .Child("users")
1791         .Child(auth.GetUid())
1792         .OnceSingleAsync<Users>()).username;
1793
1794     points2 = AppConstants.sixPoints;
1795     await firebaseClient
1796         .Child("ShoppingPoints")
1797         .Child(auth.GetUid())
1798         .PutAsync(new ShoppingPoints() { username = username, points = points2,
1799 numberOfLogs = 1, reBatCount = 1 }); ;
1800 }
1801 catch (NullReferenceException)
1802 {
1803     username = (await firebaseClient
1804         .Child("users")
1805         .Child(auth.GetUid())
1806         .OnceSingleAsync<Users>()).username;
1807
1808     points2 = AppConstants.sixPoints;
1809     await firebaseClient
1810         .Child("ShoppingPoints")
1811         .Child(auth.GetUid())
1812         .PutAsync(new ShoppingPoints() { username = username, points = points2,
1813 numberOfLogs = 1, reBatCount = 1 });
1814 }
1815 /** This function updates the points in the Shopping category by eight points. It
1816 also increments the number of logs logged in the Shopping
1817 * category by one and increments the number of times this particular action was
1818 logged by one and sends this data to Firebase.
1819 */
1820 public async void ReBagPoints()
1821 {
1822     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1823 quake-default-rtdb.firebaseio.com/");
1824     auth = DependencyService.Get<IAuth>();
1825
1826     try
1827     {
1828         username = (await firebaseClient
1829             .Child("users")
1830             .Child(auth.GetUid())
1831             .OnceSingleAsync<Users>()).username;
1832
1833         points2 = (await firebaseClient
1834             .Child("ShoppingPoints")
1835             .Child(auth.GetUid())
1836             .OnceSingleAsync<ShoppingPoints>()).points;
1837
1838         points2 = points2 + AppConstants.eightPoints;
1839
1840         numberOfLogs2 = (await firebaseClient

```

```
1839     .Child("ShoppingPoints")
1840     .Child(auth.GetUid())
1841     .OnceSingleAsync<ShoppingPoints>().numberOfLogs;
1842
1843     numberOfLogs2++;
1844
1845     clothNapkinCount2 = (await firebaseClient
1846     .Child("ShoppingPoints")
1847     .Child(auth.GetUid())
1848     .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1849
1850     clothTowelCount2 = (await firebaseClient
1851     .Child("ShoppingPoints")
1852     .Child(auth.GetUid())
1853     .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1854
1855     applianceCount2 = (await firebaseClient
1856     .Child("ShoppingPoints")
1857     .Child(auth.GetUid())
1858     .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1859
1860     productCount2 = (await firebaseClient
1861     .Child("ShoppingPoints")
1862     .Child(auth.GetUid())
1863     .OnceSingleAsync<ShoppingPoints>()).productCount;
1864
1865     toothbrushCount2 = (await firebaseClient
1866     .Child("ShoppingPoints")
1867     .Child(auth.GetUid())
1868     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
1869
1870     clothesCount2 = (await firebaseClient
1871     .Child("ShoppingPoints")
1872     .Child(auth.GetUid())
1873     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
1874
1875     foodCount2 = (await firebaseClient
1876     .Child("ShoppingPoints")
1877     .Child(auth.GetUid())
1878     .OnceSingleAsync<ShoppingPoints>()).foodCount;
1879
1880     localCount2 = (await firebaseClient
1881     .Child("ShoppingPoints")
1882     .Child(auth.GetUid())
1883     .OnceSingleAsync<ShoppingPoints>()).localCount;
1884
1885     looseLeafCount2 = (await firebaseClient
1886     .Child("ShoppingPoints")
1887     .Child(auth.GetUid())
1888     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1889
1890     organicFoodCount2 = (await firebaseClient
1891     .Child("ShoppingPoints")
1892     .Child(auth.GetUid())
1893     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1894
1895     reusableCount2 = (await firebaseClient
1896     .Child("ShoppingPoints")
1897     .Child(auth.GetUid())
1898     .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1899
```

```
1900     reBatCount2 = (await firebaseClient
1901     .Child("ShoppingPoints")
1902     .Child(auth.GetUid())
1903     .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1904
1905     reBagCount2 = (await firebaseClient
1906     .Child("ShoppingPoints")
1907     .Child(auth.GetUid())
1908     .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1909
1910     reBagCount2++;
1911
1912     await firebaseClient
1913     .Child("ShoppingPoints")
1914     .Child(auth.GetUid())
1915     .PutAsync(new ShoppingPoints()
1916     {
1917         username = username,
1918         points = points2,
1919         numberOfLogs = numberOfLogs2,
1920         clothNapkinCount = clothNapkinCount2,
1921         clothTowelCount = clothTowelCount2,
1922         applianceCount = applianceCount2,
1923         productCount = productCount2,
1924         toothbrushCount = toothbrushCount2,
1925         clothesCount = clothesCount2,
1926         foodCount = foodCount2,
1927         localCount = localCount2,
1928         looseLeafCount = looseLeafCount2,
1929         organicFoodCount = organicFoodCount2,
1930         reusableCount = reusableCount2,
1931         reBatCount = reBatCount2,
1932         reBagCount = reBagCount2,
1933     });
1934 }
1935 catch (FirebaseException)
1936 {
1937     username = (await firebaseClient
1938     .Child("users")
1939     .Child(auth.GetUid())
1940     .OnceSingleAsync<Users>()).username;
1941
1942     points2 = AppConstants.eightPoints;
1943     await firebaseClient
1944     .Child("ShoppingPoints")
1945     .Child(auth.GetUid())
1946     .PutAsync(new ShoppingPoints() { username = username, points = points2,
1947     numberOfLogs = 1, reBagCount = 1 }); ;
1948 }
1949 catch (NullReferenceException)
1950 {
1951     username = (await firebaseClient
1952     .Child("users")
1953     .Child(auth.GetUid())
1954     .OnceSingleAsync<Users>()).username;
1955
1956     points2 = AppConstants.eightPoints;
1957     await firebaseClient
1958     .Child("ShoppingPoints")
1959     .Child(auth.GetUid())
```

```
1960         .PutAsync(new ShoppingPoints() { username = username, points = points2,  
1961         numberOfLogs = 1, reBagCount = 1 });  
1962     }  
1963 }  
1964 }
```

```
1  /*! \class The TravelPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the TravelPointsUpdate ViewModel Class. It updates the data for the
7  * Travel Category of the application. The functions in this class
8  * work by reading in all the chosen data and updating the selected fields and then sending
9  * this data to back firebase.
10 *
11 */
12 using Application_Green_Quake.Models;
13 using Firebase.Database;
14 using Firebase.Database.Query;
15 using System;
16 using Xamarin.Forms;
17
18 namespace Application_Green_Quake.ViewModels
19 {
20     class TravelPointsUpdate
21     {
22         int points2 = 0;
23         int numberOfLogs2 = 0;
24         int carpoolCount2 = 0;
25         int cycleCount2 = 0;
26         int ecoCarCount2 = 0;
27         int transportCount2 = 0;
28         int walkCount2 = 0;
29
30         string username = "";
31
32         IAuth auth;
33         /** This function updates the points in the Travel category by six points. It also
34         increments the number of logs logged in the Travel
35         * category by one and increments the number of times this particular action was
36         logged by one and sends this data to Firebase.
37         */
38         public async void CarpoolPoints()
39         {
40
41             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
42             quake-default-rtdb.firebaseio.com/");
43             auth = DependencyService.Get<IAuth>();
44
45             try
46             {
47                 username = (await firebaseClient
48                 .Child("users")
49                 .Child(auth.GetUid())
50                 .OnceSingleAsync<Users>()).username;
51
52                 points2 = (await firebaseClient
53                 .Child("TravelPoints")
54                 .Child(auth.GetUid())
55                 .OnceSingleAsync<TravelPoints>()).points;
56
57                 points2 = points2 + AppConstants.sixPoints;
58
59                 numberOfLogs2 = (await firebaseClient
60                 .Child("TravelPoints")
```

```
57     .Child(auth.GetUid())
58     .OnceSingleAsync<TravelPoints>()).numberOfLogs;
59
60     numberOfLogs2++;
61
62     carpoolCount2 = (await firebaseClient
63     .Child("TravelPoints")
64     .Child(auth.GetUid())
65     .OnceSingleAsync<TravelPoints>()).carpoolCount;
66
67     carpoolCount2++;
68
69     cycleCount2 = (await firebaseClient
70     .Child("TravelPoints")
71     .Child(auth.GetUid())
72     .OnceSingleAsync<TravelPoints>()).cycleCount;
73
74     ecoCarCount2 = (await firebaseClient
75     .Child("TravelPoints")
76     .Child(auth.GetUid())
77     .OnceSingleAsync<TravelPoints>()).ecoCarCount;
78
79     transportCount2 = (await firebaseClient
80     .Child("TravelPoints")
81     .Child(auth.GetUid())
82     .OnceSingleAsync<TravelPoints>()).transportCount;
83
84     walkCount2 = (await firebaseClient
85     .Child("TravelPoints")
86     .Child(auth.GetUid())
87     .OnceSingleAsync<TravelPoints>()).walkCount;
88
89     await firebaseClient
90     .Child("TravelPoints")
91     .Child(auth.GetUid())
92     .PutAsync(new TravelPoints()
93     {
94         username = username,
95         points = points2,
96         numberOfLogs = numberOfLogs2,
97         carpoolCount = carpoolCount2,
98         cycleCount = cycleCount2,
99         ecoCarCount = ecoCarCount2,
100        transportCount = transportCount2,
101        walkCount = walkCount2,
102    });
103 }
104 catch (FirebaseException)
105 {
106     username = (await firebaseClient
107     .Child("users")
108     .Child(auth.GetUid())
109     .OnceSingleAsync<Users>()).username;
110
111     points2 = AppConstants.sixPoints;
112     await firebaseClient
113     .Child("TravelPoints")
114     .Child(auth.GetUid())
115     .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, carpoolCount = 1 }); ;
116
```



```
117     }
118     catch (NullReferenceException)
119     {
120         username = (await firebaseClient
121             .Child("users")
122             .Child(auth.GetUid())
123             .OnceSingleAsync<Users>()).username;
124
125         points2 = AppConstants.sixPoints;
126         await firebaseClient
127             .Child("TravelPoints")
128             .Child(auth.GetUid())
129             .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, carpoolCount = 1 });
130     }
131 }
132 /** This function updates the points in the Travel category by ten points. It also
increments the number of logs logged in the Travel
133 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
134 */
135 public async void CyclePoints()
136 {
137
138     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
139     auth = DependencyService.Get<IAuth>();
140
141     try
142     {
143         username = (await firebaseClient
144             .Child("users")
145             .Child(auth.GetUid())
146             .OnceSingleAsync<Users>()).username;
147
148         points2 = (await firebaseClient
149             .Child("TravelPoints")
150             .Child(auth.GetUid())
151             .OnceSingleAsync<TravelPoints>()).points;
152
153         points2 = points2 + AppConstants.tenPoints;
154
155         numberOfLogs2 = (await firebaseClient
156             .Child("TravelPoints")
157             .Child(auth.GetUid())
158             .OnceSingleAsync<TravelPoints>()).numberOfLogs;
159
160         numberOfLogs2++;
161
162         carpoolCount2 = (await firebaseClient
163             .Child("TravelPoints")
164             .Child(auth.GetUid())
165             .OnceSingleAsync<TravelPoints>()).carpoolCount;
166
167         cycleCount2 = (await firebaseClient
168             .Child("TravelPoints")
169             .Child(auth.GetUid())
170             .OnceSingleAsync<TravelPoints>()).cycleCount;
171
172         cycleCount2++;
173
174         ecoCarCount2 = (await firebaseClient
```

```

175         .Child("TravelPoints")
176         .Child(auth.GetUid())
177         .OnceSingleAsync<TravelPoints>()).ecoCarCount;
178
179         transportCount2 = (await firebaseClient
180         .Child("TravelPoints")
181         .Child(auth.GetUid())
182         .OnceSingleAsync<TravelPoints>()).transportCount;
183
184         walkCount2 = (await firebaseClient
185         .Child("TravelPoints")
186         .Child(auth.GetUid())
187         .OnceSingleAsync<TravelPoints>()).walkCount;
188
189         await firebaseClient
190         .Child("TravelPoints")
191         .Child(auth.GetUid())
192         .PutAsync(new TravelPoints()
193         {
194             username = username,
195             points = points2,
196             numberOfLogs = numberOfLogs2,
197             carpoolCount = carpoolCount2,
198             cycleCount = cycleCount2,
199             ecoCarCount = ecoCarCount2,
200             transportCount = transportCount2,
201             walkCount = walkCount2,
202         });
203     }
204     catch (FirebaseException)
205     {
206         username = (await firebaseClient
207         .Child("users")
208         .Child(auth.GetUid())
209         .OnceSingleAsync<Users>()).username;
210
211         points2 = AppConstants.tenPoints;
212         await firebaseClient
213         .Child("TravelPoints")
214         .Child(auth.GetUid())
215         .PutAsync(new TravelPoints() { username = username, points = points2,
216         numberOfLogs = 1, cycleCount = 1 }); ;
217     }
218     catch (NullReferenceException)
219     {
220         username = (await firebaseClient
221         .Child("users")
222         .Child(auth.GetUid())
223         .OnceSingleAsync<Users>()).username;
224
225         points2 = AppConstants.tenPoints;
226         await firebaseClient
227         .Child("TravelPoints")
228         .Child(auth.GetUid())
229         .PutAsync(new TravelPoints() { username = username, points = points2,
230         numberOfLogs = 1, cycleCount = 1 }); ;
231     }
232     /** This function updates the points in the Travel category by ten points. It also
233     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.
234     */
235     public async void EcoCarPoints()
236     {
237
238         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
239         auth = DependencyService.Get<IAuth>();
240
241         try
242         {
243             username = (await firebaseClient
244                 .Child("users")
245                 .Child(auth.GetUid())
246                 .OnceSingleAsync<Users>()).username;
247
248             points2 = (await firebaseClient
249                 .Child("TravelPoints")
250                 .Child(auth.GetUid())
251                 .OnceSingleAsync<TravelPoints>()).points;
252
253             points2 = points2 + AppConstants.tenPoints;
254
255             numberOfLogs2 = (await firebaseClient
256                 .Child("TravelPoints")
257                 .Child(auth.GetUid())
258                 .OnceSingleAsync<TravelPoints>()).numberOfLogs;
259
260             numberOfLogs2++;
261
262             carpoolCount2 = (await firebaseClient
263                 .Child("TravelPoints")
264                 .Child(auth.GetUid())
265                 .OnceSingleAsync<TravelPoints>()).carpoolCount;
266
267             cycleCount2 = (await firebaseClient
268                 .Child("TravelPoints")
269                 .Child(auth.GetUid())
270                 .OnceSingleAsync<TravelPoints>()).cycleCount;
271
272             ecoCarCount2 = (await firebaseClient
273                 .Child("TravelPoints")
274                 .Child(auth.GetUid())
275                 .OnceSingleAsync<TravelPoints>()).ecoCarCount;
276
277             ecoCarCount2++;
278
279             transportCount2 = (await firebaseClient
280                 .Child("TravelPoints")
281                 .Child(auth.GetUid())
282                 .OnceSingleAsync<TravelPoints>()).transportCount;
283
284             walkCount2 = (await firebaseClient
285                 .Child("TravelPoints")
286                 .Child(auth.GetUid())
287                 .OnceSingleAsync<TravelPoints>()).walkCount;
288
289             await firebaseClient
290                 .Child("TravelPoints")
291                 .Child(auth.GetUid())
292                 .PutAsync(new TravelPoints()
```

```

293     {
294         username = username,
295         points = points2,
296         numberOfLogs = numberOfLogs2,
297         carpoolCount = carpoolCount2,
298         cycleCount = cycleCount2,
299         ecoCarCount = ecoCarCount2,
300         transportCount = transportCount2,
301         walkCount = walkCount2,
302     });
303 }
304 catch (FirebaseException)
305 {
306     username = (await firebaseClient
307         .Child("users")
308         .Child(auth.GetUid())
309         .OnceSingleAsync<Users>()).username;
310
311     points2 = AppConstants.tenPoints;
312     await firebaseClient
313         .Child("TravelPoints")
314         .Child(auth.GetUid())
315         .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, ecoCarCount = 1 }); ;
316 }
317 catch (NullReferenceException)
318 {
319     username = (await firebaseClient
320         .Child("users")
321         .Child(auth.GetUid())
322         .OnceSingleAsync<Users>()).username;
323
324     points2 = AppConstants.tenPoints;
325     await firebaseClient
326         .Child("TravelPoints")
327         .Child(auth.GetUid())
328         .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, ecoCarCount = 1 }); ;
329 }
330 }
331 }
332 /** This function updates the points in the Travel category by eight points. It
also increments the number of logs logged in the Travel
333 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
334 */
335 public async void TransportPoints()
336 {
337
338     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
339     auth = DependencyService.Get<IAuth>();
340
341     try
342     {
343         username = (await firebaseClient
344             .Child("users")
345             .Child(auth.GetUid())
346             .OnceSingleAsync<Users>()).username;
347
348         points2 = (await firebaseClient
349             .Child("TravelPoints")

```

```
350     .Child(auth.GetUid())
351     .OnceSingleAsync<TravelPoints>()).points;
352
353     points2 = points2 + AppConstants.eightPoints;
354
355     numberOfLogs2 = (await firebaseClient
356     .Child("TravelPoints")
357     .Child(auth.GetUid())
358     .OnceSingleAsync<TravelPoints>()).numberOfLogs;
359
360     numberOfLogs2++;
361
362     carpoolCount2 = (await firebaseClient
363     .Child("TravelPoints")
364     .Child(auth.GetUid())
365     .OnceSingleAsync<TravelPoints>()).carpoolCount;
366
367     cycleCount2 = (await firebaseClient
368     .Child("TravelPoints")
369     .Child(auth.GetUid())
370     .OnceSingleAsync<TravelPoints>()).cycleCount;
371
372     ecoCarCount2 = (await firebaseClient
373     .Child("TravelPoints")
374     .Child(auth.GetUid())
375     .OnceSingleAsync<TravelPoints>()).ecoCarCount;
376
377     transportCount2 = (await firebaseClient
378     .Child("TravelPoints")
379     .Child(auth.GetUid())
380     .OnceSingleAsync<TravelPoints>()).transportCount;
381
382     transportCount2++;
383
384     walkCount2 = (await firebaseClient
385     .Child("TravelPoints")
386     .Child(auth.GetUid())
387     .OnceSingleAsync<TravelPoints>()).walkCount;
388
389     await firebaseClient
390     .Child("TravelPoints")
391     .Child(auth.GetUid())
392     .PutAsync(new TravelPoints()
393     {
394         username = username,
395         points = points2,
396         numberOfLogs = numberOfLogs2,
397         carpoolCount = carpoolCount2,
398         cycleCount = cycleCount2,
399         ecoCarCount = ecoCarCount2,
400         transportCount = transportCount2,
401         walkCount = walkCount2,
402     });
403 }
404 catch (FirebaseException)
405 {
406     username = (await firebaseClient
407     .Child("users")
408     .Child(auth.GetUid())
409     .OnceSingleAsync<Users>()).username;
410
```

```
411         points2 = AppConstants.eightPoints;
412         await firebaseClient
413             .Child("TravelPoints")
414             .Child(auth.GetUid())
415             .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, transportCount = 1 }); ;
416     }
417     catch (NullReferenceException)
418     {
419         username = (await firebaseClient
420             .Child("users")
421             .Child(auth.GetUid())
422             .OnceSingleAsync<Users>()).username;
423
424         points2 = AppConstants.eightPoints;
425         await firebaseClient
426             .Child("TravelPoints")
427             .Child(auth.GetUid())
428             .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, transportCount = 1 }); ;
429     }
430 }
431
432 /** This function updates the points in the Travel category by ten points. It also
increments the number of logs logged in the Travel
433 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
434 */
435 public async void WalkPoints()
436 {
437
438     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
439     auth = DependencyService.Get<IAuth>();
440
441     try
442     {
443         username = (await firebaseClient
444             .Child("users")
445             .Child(auth.GetUid())
446             .OnceSingleAsync<Users>()).username;
447
448         points2 = (await firebaseClient
449             .Child("TravelPoints")
450             .Child(auth.GetUid())
451             .OnceSingleAsync<TravelPoints>()).points;
452
453         points2 = points2 + AppConstants.tenPoints;
454
455         numberOfLogs2 = (await firebaseClient
456             .Child("TravelPoints")
457             .Child(auth.GetUid())
458             .OnceSingleAsync<TravelPoints>()).numberOfLogs;
459
460         numberOfLogs2++;
461
462         carpoolCount2 = (await firebaseClient
463             .Child("TravelPoints")
464             .Child(auth.GetUid())
465             .OnceSingleAsync<TravelPoints>()).carpoolCount;
466
467         cycleCount2 = (await firebaseClient
```

```

468     .Child("TravelPoints")
469     .Child(auth.GetUid())
470     .OnceSingleAsync<TravelPoints>()).cycleCount;
471
472     ecoCarCount2 = (await firebaseClient
473     .Child("TravelPoints")
474     .Child(auth.GetUid())
475     .OnceSingleAsync<TravelPoints>()).ecoCarCount;
476
477     transportCount2 = (await firebaseClient
478     .Child("TravelPoints")
479     .Child(auth.GetUid())
480     .OnceSingleAsync<TravelPoints>()).transportCount;
481
482     walkCount2 = (await firebaseClient
483     .Child("TravelPoints")
484     .Child(auth.GetUid())
485     .OnceSingleAsync<TravelPoints>()).walkCount;
486
487     walkCount2++;
488
489     await firebaseClient
490     .Child("TravelPoints")
491     .Child(auth.GetUid())
492     .PutAsync(new TravelPoints()
493     {
494         username = username,
495         points = points2,
496         numberOfLogs = numberOfLogs2,
497         carpoolCount = carpoolCount2,
498         cycleCount = cycleCount2,
499         ecoCarCount = ecoCarCount2,
500         transportCount = transportCount2,
501         walkCount = walkCount2,
502     });
503 }
504 catch (FirebaseException)
505 {
506     username = (await firebaseClient
507     .Child("users")
508     .Child(auth.GetUid())
509     .OnceSingleAsync<Users>()).username;
510
511     points2 = AppConstants.tenPoints;
512     await firebaseClient
513     .Child("TravelPoints")
514     .Child(auth.GetUid())
515     .PutAsync(new TravelPoints() { username = username, points = points2,
516 numberOfLogs = 1, walkCount = 1 }); ;
517 }
518 catch (NullReferenceException)
519 {
520     username = (await firebaseClient
521     .Child("users")
522     .Child(auth.GetUid())
523     .OnceSingleAsync<Users>()).username;
524
525     points2 = AppConstants.tenPoints;
526     await firebaseClient
527     .Child("TravelPoints")

```

```
528         .Child(auth.GetUid())
529         .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, walkCount = 1 });
530     }
531 }
532 }
533 }
```



```
1  /*! \class The WastePointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the WastePointsUpdate ViewModel Class. It updates the data for the
7  * Waste Category of the application. The functions in this class
8  * work by reading in all the chosen data and updating the selected fields and then sending
9  * this data to back firebase.
10 *
11 */
12 using Application_Green_Quake.Models;
13 using Firebase.Database;
14 using Firebase.Database.Query;
15 using System;
16 using Xamarin.Forms;
17
18 namespace Application_Green_Quake.ViewModels
19 {
20     class WastePointsUpdate
21     {
22         int points2 = 0;
23         int numberOfLogs2 = 0;
24         int billsCount2 = 0;
25         int compostCount2 = 0;
26         int setUpRecyclingBinCount2 = 0;
27         int bioBinBagsCount2 = 0;
28         int recyclingBinCount2 = 0;
29
30         string username = "";
31
32         IAuth auth;
33         /** This function updates the points in the Waste category by four points. It also
34         increments the number of logs logged in the Waste
35         * category by one and increments the number of times this particular action was
36         logged by one and sends this data to Firebase.
37         */
38         public async void BillsPoints()
39         {
40
41             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
42             quake-default-rtdb.firebaseio.com/");
43             auth = DependencyService.Get<IAuth>();
44
45             try
46             {
47                 username = (await firebaseClient
48                 .Child("users")
49                 .Child(auth.GetUid())
50                 .OnceSingleAsync<Users>()).username;
51
52                 points2 = (await firebaseClient
53                 .Child("WastePoints")
54                 .Child(auth.GetUid())
55                 .OnceSingleAsync<WastePoints>()).points;
56
57                 points2 = points2 + AppConstants.fourPoints;
58
59                 numberOfLogs2 = (await firebaseClient
60                 .Child("WastePoints")
```

```
57     .Child(auth.GetUid())
58     .OnceSingleAsync<WastePoints>()).numberOfLogs;
59
60     numberOfLogs2++;
61
62     billsCount2 = (await firebaseClient
63     .Child("WastePoints")
64     .Child(auth.GetUid())
65     .OnceSingleAsync<WastePoints>()).billsCount;
66
67     billsCount2++;
68
69     compostCount2 = (await firebaseClient
70     .Child("WastePoints")
71     .Child(auth.GetUid())
72     .OnceSingleAsync<WastePoints>()).compostCount;
73
74     setUpRecyclingBinCount2 = (await firebaseClient
75     .Child("WastePoints")
76     .Child(auth.GetUid())
77     .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
78
79     bioBinBagsCount2 = (await firebaseClient
80     .Child("WastePoints")
81     .Child(auth.GetUid())
82     .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
83
84     recyclingBinCount2 = (await firebaseClient
85     .Child("WastePoints")
86     .Child(auth.GetUid())
87     .OnceSingleAsync<WastePoints>()).recyclingBinCount;
88
89     await firebaseClient
90     .Child("WastePoints")
91     .Child(auth.GetUid())
92     .PutAsync(new WastePoints()
93     {
94         username = username,
95         points = points2,
96         numberOfLogs = numberOfLogs2,
97         billsCount = billsCount2,
98         compostCount = compostCount2,
99         setUpRecyclingBinCount = setUpRecyclingBinCount2,
100        bioBinBagsCount = bioBinBagsCount2,
101        recyclingBinCount = recyclingBinCount2,
102    });
103 }
104 catch (FirebaseException)
105 {
106     username = (await firebaseClient
107     .Child("users")
108     .Child(auth.GetUid())
109     .OnceSingleAsync<Users>()).username;
110
111     points2 = AppConstants.fourPoints;
112     await firebaseClient
113     .Child("WastePoints")
114     .Child(auth.GetUid())
115     .PutAsync(new WastePoints() { username = username, points = points2,
116     numberOfLogs = 1, billsCount = 1 }); ;
```

```
117     }
118     catch (NullReferenceException)
119     {
120         username = (await firebaseClient
121             .Child("users")
122             .Child(auth.GetUid())
123             .OnceSingleAsync<Users>()).username;
124
125         points2 = AppConstants.fourPoints;
126         await firebaseClient
127             .Child("WastePoints")
128             .Child(auth.GetUid())
129             .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, billsCount = 1 });
130     }
131 }
132 /** This function updates the points in the Waste category by six points. It also
increments the number of logs logged in the Waste
133 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
134 */
135 public async void CompostPoints()
136 {
137
138     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
139     auth = DependencyService.Get<IAuth>();
140
141     try
142     {
143         username = (await firebaseClient
144             .Child("users")
145             .Child(auth.GetUid())
146             .OnceSingleAsync<Users>()).username;
147
148         points2 = (await firebaseClient
149             .Child("WastePoints")
150             .Child(auth.GetUid())
151             .OnceSingleAsync<WastePoints>()).points;
152
153         points2 = points2 + AppConstants.sixPoints;
154
155         numberOfLogs2 = (await firebaseClient
156             .Child("WastePoints")
157             .Child(auth.GetUid())
158             .OnceSingleAsync<WastePoints>()).numberOfLogs;
159
160         numberOfLogs2++;
161
162         billsCount2 = (await firebaseClient
163             .Child("WastePoints")
164             .Child(auth.GetUid())
165             .OnceSingleAsync<WastePoints>()).billsCount;
166
167         compostCount2 = (await firebaseClient
168             .Child("WastePoints")
169             .Child(auth.GetUid())
170             .OnceSingleAsync<WastePoints>()).compostCount;
171
172         compostCount2++;
173
174         setUpRecyclingBinCount2 = (await firebaseClient
```

```

175         .Child("WastePoints")
176         .Child(auth.GetUid())
177         .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
178
179         bioBinBagsCount2 = (await firebaseClient
180         .Child("WastePoints")
181         .Child(auth.GetUid())
182         .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
183
184         recyclingBinCount2 = (await firebaseClient
185         .Child("WastePoints")
186         .Child(auth.GetUid())
187         .OnceSingleAsync<WastePoints>()).recyclingBinCount;
188
189         await firebaseClient
190         .Child("WastePoints")
191         .Child(auth.GetUid())
192         .PutAsync(new WastePoints()
193         {
194             username = username,
195             points = points2,
196             numberOfLogs = numberOfLogs2,
197             billsCount = billsCount2,
198             compostCount = compostCount2,
199             setUpRecyclingBinCount = setUpRecyclingBinCount2,
200             bioBinBagsCount = bioBinBagsCount2,
201             recyclingBinCount = recyclingBinCount2,
202         });
203     }
204     catch (FirebaseException)
205     {
206         username = (await firebaseClient
207         .Child("users")
208         .Child(auth.GetUid())
209         .OnceSingleAsync<Users>()).username;
210
211         points2 = AppConstants.sixPoints;
212         await firebaseClient
213         .Child("WastePoints")
214         .Child(auth.GetUid())
215         .PutAsync(new WastePoints() { username = username, points = points2,
216         numberOfLogs = 1, compostCount = 1 }); ;
217     }
218     catch (NullReferenceException)
219     {
220         username = (await firebaseClient
221         .Child("users")
222         .Child(auth.GetUid())
223         .OnceSingleAsync<Users>()).username;
224
225         points2 = AppConstants.sixPoints;
226         await firebaseClient
227         .Child("WastePoints")
228         .Child(auth.GetUid())
229         .PutAsync(new WastePoints() { username = username, points = points2,
230         numberOfLogs = 1, compostCount = 1 }); ;
231     }
232     /** This function updates the points in the Waste category by ten points. It also
233     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.
234     */
235     public async void SetUpRecyclingBinPoints()
236     {
237
238         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
239         auth = DependencyService.Get<IAuth>();
240
241         try
242         {
243             username = (await firebaseClient
244                 .Child("users")
245                 .Child(auth.GetUid())
246                 .OnceSingleAsync<Users>()).username;
247
248             points2 = (await firebaseClient
249                 .Child("WastePoints")
250                 .Child(auth.GetUid())
251                 .OnceSingleAsync<WastePoints>()).points;
252
253             points2 = points2 + AppConstants.tenPoints;
254
255             numberOfLogs2 = (await firebaseClient
256                 .Child("WastePoints")
257                 .Child(auth.GetUid())
258                 .OnceSingleAsync<WastePoints>()).numberOfLogs;
259
260             numberOfLogs2++;
261
262             billsCount2 = (await firebaseClient
263                 .Child("WastePoints")
264                 .Child(auth.GetUid())
265                 .OnceSingleAsync<WastePoints>()).billsCount;
266
267             compostCount2 = (await firebaseClient
268                 .Child("WastePoints")
269                 .Child(auth.GetUid())
270                 .OnceSingleAsync<WastePoints>()).compostCount;
271
272             setUpRecyclingBinCount2 = (await firebaseClient
273                 .Child("WastePoints")
274                 .Child(auth.GetUid())
275                 .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
276
277             setUpRecyclingBinCount2++;
278
279             bioBinBagsCount2 = (await firebaseClient
280                 .Child("WastePoints")
281                 .Child(auth.GetUid())
282                 .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
283
284             recyclingBinCount2 = (await firebaseClient
285                 .Child("WastePoints")
286                 .Child(auth.GetUid())
287                 .OnceSingleAsync<WastePoints>()).recyclingBinCount;
288
289             await firebaseClient
290                 .Child("WastePoints")
291                 .Child(auth.GetUid())
292                 .PutAsync(new WastePoints()
```

```

293     {
294         username = username,
295         points = points2,
296         numberOfLogs = numberOfLogs2,
297         billsCount = billsCount2,
298         compostCount = compostCount2,
299         setUpRecyclingBinCount = setUpRecyclingBinCount2,
300         bioBinBagsCount = bioBinBagsCount2,
301         recyclingBinCount = recyclingBinCount2,
302     });
303 }
304 catch (FirebaseException)
305 {
306     username = (await firebaseClient
307         .Child("users")
308         .Child(auth.GetUid())
309         .OnceSingleAsync<Users>()).username;
310
311     points2 = AppConstants.tenPoints;
312     await firebaseClient
313         .Child("WastePoints")
314         .Child(auth.GetUid())
315         .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, setUpRecyclingBinCount = 1 }); ;
316
317 }
318 catch (NullReferenceException)
319 {
320     username = (await firebaseClient
321         .Child("users")
322         .Child(auth.GetUid())
323         .OnceSingleAsync<Users>()).username;
324
325     points2 = AppConstants.tenPoints;
326     await firebaseClient
327         .Child("WastePoints")
328         .Child(auth.GetUid())
329         .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, setUpRecyclingBinCount = 1 }); ;
330     }
331 }
332 /** This function updates the points in the Waste category by four points. It also
increments the number of logs logged in the Waste
333 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
334 */
335 public async void BioBinBagPoints()
336 {
337
338     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
339     auth = DependencyService.Get<IAuth>();
340
341     try
342     {
343         username = (await firebaseClient
344             .Child("users")
345             .Child(auth.GetUid())
346             .OnceSingleAsync<Users>()).username;
347
348         points2 = (await firebaseClient
349             .Child("WastePoints")

```

```
350     .Child(auth.GetUid())
351     .OnceSingleAsync<WastePoints>()).points;
352
353     points2 = points2 + AppConstants.fourPoints;
354
355     numberOfLogs2 = (await firebaseClient
356     .Child("WastePoints")
357     .Child(auth.GetUid())
358     .OnceSingleAsync<WastePoints>()).numberOfLogs;
359
360     numberOfLogs2++;
361
362     billsCount2 = (await firebaseClient
363     .Child("WastePoints")
364     .Child(auth.GetUid())
365     .OnceSingleAsync<WastePoints>()).billsCount;
366
367     compostCount2 = (await firebaseClient
368     .Child("WastePoints")
369     .Child(auth.GetUid())
370     .OnceSingleAsync<WastePoints>()).compostCount;
371
372     setUpRecyclingBinCount2 = (await firebaseClient
373     .Child("WastePoints")
374     .Child(auth.GetUid())
375     .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
376
377     bioBinBagsCount2 = (await firebaseClient
378     .Child("WastePoints")
379     .Child(auth.GetUid())
380     .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
381
382     bioBinBagsCount2++;
383
384     recyclingBinCount2 = (await firebaseClient
385     .Child("WastePoints")
386     .Child(auth.GetUid())
387     .OnceSingleAsync<WastePoints>()).recyclingBinCount;
388
389     await firebaseClient
390     .Child("WastePoints")
391     .Child(auth.GetUid())
392     .PutAsync(new WastePoints()
393     {
394         username = username,
395         points = points2,
396         numberOfLogs = numberOfLogs2,
397         billsCount = billsCount2,
398         compostCount = compostCount2,
399         setUpRecyclingBinCount = setUpRecyclingBinCount2,
400         bioBinBagsCount = bioBinBagsCount2,
401         recyclingBinCount = recyclingBinCount2,
402     });
403 }
404 catch (FirebaseException)
405 {
406     username = (await firebaseClient
407     .Child("users")
408     .Child(auth.GetUid())
409     .OnceSingleAsync<Users>()).username;
410
```

```

411         points2 = AppConstants.fourPoints;
412         await firebaseClient
413             .Child("WastePoints")
414             .Child(auth.GetUid())
415             .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, bioBinBagsCount = 1 }); ;
416
417     }
418     catch (NullReferenceException)
419     {
420         username = (await firebaseClient
421             .Child("users")
422             .Child(auth.GetUid())
423             .OnceSingleAsync<Users>()).username;
424
425         points2 = AppConstants.fourPoints;
426         await firebaseClient
427             .Child("WastePoints")
428             .Child(auth.GetUid())
429             .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, bioBinBagsCount = 1 }); ;
430     }
431 }
432 /** This function updates the points in the Waste category by six points. It also
increments the number of logs logged in the Waste
433 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
434 */
435 public async void RecyclingBinPoints()
436 {
437
438     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
439     auth = DependencyService.Get<IAuth>();
440
441     try
442     {
443         username = (await firebaseClient
444             .Child("users")
445             .Child(auth.GetUid())
446             .OnceSingleAsync<Users>()).username;
447
448         points2 = (await firebaseClient
449             .Child("WastePoints")
450             .Child(auth.GetUid())
451             .OnceSingleAsync<WastePoints>()).points;
452
453         points2 = points2 + AppConstants.sixPoints;
454
455         numberOfLogs2 = (await firebaseClient
456             .Child("WastePoints")
457             .Child(auth.GetUid())
458             .OnceSingleAsync<WastePoints>()).numberOfLogs;
459
460         numberOfLogs2++;
461
462         billsCount2 = (await firebaseClient
463             .Child("WastePoints")
464             .Child(auth.GetUid())
465             .OnceSingleAsync<WastePoints>()).billsCount;
466
467         compostCount2 = (await firebaseClient

```



```

468     .Child("WastePoints")
469     .Child(auth.GetUid())
470     .OnceSingleAsync<WastePoints>()).compostCount;
471
472     setUpRecyclingBinCount2 = (await firebaseClient
473     .Child("WastePoints")
474     .Child(auth.GetUid())
475     .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
476
477     bioBinBagsCount2 = (await firebaseClient
478     .Child("WastePoints")
479     .Child(auth.GetUid())
480     .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
481
482     recyclingBinCount2 = (await firebaseClient
483     .Child("WastePoints")
484     .Child(auth.GetUid())
485     .OnceSingleAsync<WastePoints>()).recyclingBinCount;
486
487     recyclingBinCount2++;
488
489     await firebaseClient
490     .Child("WastePoints")
491     .Child(auth.GetUid())
492     .PutAsync(new WastePoints()
493     {
494         username = username,
495         points = points2,
496         numberOfLogs = numberOfLogs2,
497         billsCount = billsCount2,
498         compostCount = compostCount2,
499         setUpRecyclingBinCount = setUpRecyclingBinCount2,
500         bioBinBagsCount = bioBinBagsCount2,
501         recyclingBinCount = recyclingBinCount2,
502     });
503 }
504 catch (FirebaseException)
505 {
506     username = (await firebaseClient
507     .Child("users")
508     .Child(auth.GetUid())
509     .OnceSingleAsync<Users>()).username;
510
511     points2 = AppConstants.sixPoints;
512     await firebaseClient
513     .Child("WastePoints")
514     .Child(auth.GetUid())
515     .PutAsync(new WastePoints() { username = username, points = points2,
516     numberOfLogs = 1, recyclingBinCount = 1 }); ;
517 }
518 catch (NullReferenceException)
519 {
520     username = (await firebaseClient
521     .Child("users")
522     .Child(auth.GetUid())
523     .OnceSingleAsync<Users>()).username;
524
525     points2 = AppConstants.sixPoints;
526     await firebaseClient
527     .Child("WastePoints")

```

```
528         .Child(auth.GetUid())
529         .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, recyclingBinCount = 1 });
530     }
531 }
532 }
533 }
```

```

1  /*! \class The WaterPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the WaterPointsUpdate ViewModel Class. It updates the data for the
   Water Category of the application. The functions in this class
7  * work by reading in all the chosen data and updating the selected fields and then sending
   this data to back firebase.
8  *
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class WaterPointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int cisternCount2 = 0;
23         int rainBarrel2 = 0;
24         int reWater2 = 0;
25         int showerBucket2 = 0;
26         int wSShowerHead2 = 0;
27
28         string username = "";
29
30         IAuth auth;
31         /** This function updates the points in the Water category by ten points. It also
   increments the number of logs logged in the Water
32         * category by one and increments the number of times this particular action was
   logged by one and sends this data to Firebase.
33         */
34         public async void CisternPoints()
35         {
36
37             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
   quake-default-rtdb.firebaseio.com/");
38             auth = DependencyService.Get<IAuth>();
39
40             try
41             {
42                 username = (await firebaseClient
43                     .Child("users")
44                     .Child(auth.GetUid())
45                     .OnceSingleAsync<Users>()).username;
46
47                 points2 = (await firebaseClient
48                     .Child("WaterPoints")
49                     .Child(auth.GetUid())
50                     .OnceSingleAsync<WaterPoints>()).points;
51
52                 points2 = points2 + AppConstants.tenPoints;
53
54                 numberOfLogs2 = (await firebaseClient
55                     .Child("WaterPoints")
56                     .Child(auth.GetUid())

```

```
57         .OnceSingleAsync<WaterPoints>()).numberOfLogs;
58
59         numberOfLogs2++;
60
61         cisternCount2 = (await firebaseClient
62             .Child("WaterPoints")
63             .Child(auth.GetUid())
64             .OnceSingleAsync<WaterPoints>()).cisternCount;
65
66         cisternCount2++;
67
68         rainBarrel2 = (await firebaseClient
69             .Child("WaterPoints")
70             .Child(auth.GetUid())
71             .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
72
73         reWater2 = (await firebaseClient
74             .Child("WaterPoints")
75             .Child(auth.GetUid())
76             .OnceSingleAsync<WaterPoints>()).reWaterCount;
77
78         showerBucket2 = (await firebaseClient
79             .Child("WaterPoints")
80             .Child(auth.GetUid())
81             .OnceSingleAsync<WaterPoints>()).showerBucketCount;
82
83         wSShowerHead2 = (await firebaseClient
84             .Child("WaterPoints")
85             .Child(auth.GetUid())
86             .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
87
88         await firebaseClient
89             .Child("WaterPoints")
90             .Child(auth.GetUid())
91             .PutAsync(new WaterPoints()
92             {
93                 username = username,
94                 points = points2,
95                 numberOfLogs = numberOfLogs2,
96                 cisternCount = cisternCount2,
97                 rainBarrelCount = rainBarrel2,
98                 reWaterCount = reWater2,
99                 showerBucketCount = showerBucket2,
100                wSShowerHeadCount = wSShowerHead2,
101            });
102     }
103     catch (FirebaseException)
104     {
105         username = (await firebaseClient
106             .Child("users")
107             .Child(auth.GetUid())
108             .OnceSingleAsync<Users>()).username;
109
110         points2 = AppConstants.tenPoints;
111         await firebaseClient
112             .Child("WaterPoints")
113             .Child(auth.GetUid())
114             .PutAsync(new WaterPoints() { username = username, points = points2,
115 numberOfLogs = 1, cisternCount = 1 }); ;
116     }
```

```
117     catch (NullReferenceException)
118     {
119         username = (await firebaseClient
120             .Child("users")
121             .Child(auth.GetUid())
122             .OnceSingleAsync<Users>()).username;
123
124         points2 = AppConstants.tenPoints;
125         await firebaseClient
126             .Child("WaterPoints")
127             .Child(auth.GetUid())
128             .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, cisternCount = 1 });
129     }
130 }
131 /** This function updates the points in the Water category by ten points. It also
increments the number of logs logged in the Water
132 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
133 */
134 public async void BarrelPoints()
135 {
136
137     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
138     auth = DependencyService.Get<IAuth>();
139
140     try
141     {
142         username = (await firebaseClient
143             .Child("users")
144             .Child(auth.GetUid())
145             .OnceSingleAsync<Users>()).username;
146
147         points2 = (await firebaseClient
148             .Child("WaterPoints")
149             .Child(auth.GetUid())
150             .OnceSingleAsync<WaterPoints>()).points;
151
152         points2 = points2 + AppConstants.tenPoints;
153
154         numberOfLogs2 = (await firebaseClient
155             .Child("WaterPoints")
156             .Child(auth.GetUid())
157             .OnceSingleAsync<WaterPoints>()).numberOfLogs;
158
159         numberOfLogs2++;
160
161         cisternCount2 = (await firebaseClient
162             .Child("WaterPoints")
163             .Child(auth.GetUid())
164             .OnceSingleAsync<WaterPoints>()).cisternCount;
165
166         rainBarrel2 = (await firebaseClient
167             .Child("WaterPoints")
168             .Child(auth.GetUid())
169             .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
170
171         rainBarrel2++;
172
173         reWater2 = (await firebaseClient
174             .Child("WaterPoints")
```

```

175         .Child(auth.GetUid())
176         .OnceSingleAsync<WaterPoints>()).reWaterCount;
177
178         showerBucket2 = (await firebaseClient
179         .Child("WaterPoints")
180         .Child(auth.GetUid())
181         .OnceSingleAsync<WaterPoints>()).showerBucketCount;
182
183         wSShowerHead2 = (await firebaseClient
184         .Child("WaterPoints")
185         .Child(auth.GetUid())
186         .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
187
188         await firebaseClient
189         .Child("WaterPoints")
190         .Child(auth.GetUid())
191         .PutAsync(new WaterPoints()
192         {
193             username = username,
194             points = points2,
195             numberOfLogs = numberOfLogs2,
196             cisternCount = cisternCount2,
197             rainBarrelCount = rainBarrel2,
198             reWaterCount = reWater2,
199             showerBucketCount = showerBucket2,
200             wSShowerHeadCount = wSShowerHead2,
201         });
202     }
203     catch (FirebaseException)
204     {
205         username = (await firebaseClient
206         .Child("users")
207         .Child(auth.GetUid())
208         .OnceSingleAsync<Users>()).username;
209
210         points2 = AppConstants.tenPoints;
211         await firebaseClient
212         .Child("WaterPoints")
213         .Child(auth.GetUid())
214         .PutAsync(new WaterPoints() { username = username, points = points2,
215         numberOfLogs = 1, rainBarrelCount = 1 }); ;
216     }
217     catch (NullReferenceException)
218     {
219         username = (await firebaseClient
220         .Child("users")
221         .Child(auth.GetUid())
222         .OnceSingleAsync<Users>()).username;
223
224         points2 = AppConstants.tenPoints;
225         await firebaseClient
226         .Child("WaterPoints")
227         .Child(auth.GetUid())
228         .PutAsync(new WaterPoints() { username = username, points = points2,
229         numberOfLogs = 1, rainBarrelCount = 1 });
230     }
231     /** This function updates the points in the Water category by eight points. It also
232     increments the number of logs logged in the Water
233     * category by one and increments the number of times this particular action was
234     logged by one and sends this data to Firebase.

```

```
233     */
234     public async void ReWaterPoints()
235     {
236
237         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
238         auth = DependencyService.Get<IAuth>();
239
240         try
241         {
242             username = (await firebaseClient
243                 .Child("users")
244                 .Child(auth.GetUid())
245                 .OnceSingleAsync<Users>()).username;
246
247             points2 = (await firebaseClient
248                 .Child("WaterPoints")
249                 .Child(auth.GetUid())
250                 .OnceSingleAsync<WaterPoints>()).points;
251
252             points2 = points2 + AppConstants.eightPoints;
253
254             numberOfLogs2 = (await firebaseClient
255                 .Child("WaterPoints")
256                 .Child(auth.GetUid())
257                 .OnceSingleAsync<WaterPoints>()).numberOfLogs;
258
259             numberOfLogs2++;
260
261             cisternCount2 = (await firebaseClient
262                 .Child("WaterPoints")
263                 .Child(auth.GetUid())
264                 .OnceSingleAsync<WaterPoints>()).cisternCount;
265
266             rainBarrel2 = (await firebaseClient
267                 .Child("WaterPoints")
268                 .Child(auth.GetUid())
269                 .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
270
271             reWater2 = (await firebaseClient
272                 .Child("WaterPoints")
273                 .Child(auth.GetUid())
274                 .OnceSingleAsync<WaterPoints>()).reWaterCount;
275
276             reWater2++;
277
278             showerBucket2 = (await firebaseClient
279                 .Child("WaterPoints")
280                 .Child(auth.GetUid())
281                 .OnceSingleAsync<WaterPoints>()).showerBucketCount;
282
283             wSShowerHead2 = (await firebaseClient
284                 .Child("WaterPoints")
285                 .Child(auth.GetUid())
286                 .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
287
288             await firebaseClient
289                 .Child("WaterPoints")
290                 .Child(auth.GetUid())
291                 .PutAsync(new WaterPoints()
292                 {
```

```

293         username = username,
294         points = points2,
295         numberOfLogs = numberOfLogs2,
296         cisternCount = cisternCount2,
297         rainBarrelCount = rainBarrel2,
298         reWaterCount = reWater2,
299         showerBucketCount = showerBucket2,
300         wSShowerHeadCount = wSShowerHead2,
301     });
302 }
303 catch (FirebaseException)
304 {
305     username = (await firebaseClient
306         .Child("users")
307         .Child(auth.GetUid())
308         .OnceSingleAsync<Users>()).username;
309
310     points2 = AppConstants.eightPoints;
311     await firebaseClient
312         .Child("WaterPoints")
313         .Child(auth.GetUid())
314         .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, reWaterCount = 1 }); ;
315 }
316 catch (NullReferenceException)
317 {
318     username = (await firebaseClient
319         .Child("users")
320         .Child(auth.GetUid())
321         .OnceSingleAsync<Users>()).username;
322
323     points2 = AppConstants.eightPoints;
324     await firebaseClient
325         .Child("WaterPoints")
326         .Child(auth.GetUid())
327         .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, reWaterCount = 1 });
328 }
329 }
330
331 /** This function updates the points in the Water category by eight points. It also
increments the number of logs logged in the Water
332 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
333 */
334 public async void ShowerBucketPoints()
335 {
336
337     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
338     auth = DependencyService.Get<IAuth>();
339
340     try
341     {
342         username = (await firebaseClient
343             .Child("users")
344             .Child(auth.GetUid())
345             .OnceSingleAsync<Users>()).username;
346
347         points2 = (await firebaseClient
348             .Child("WaterPoints")
349             .Child(auth.GetUid())

```



```
350     .OnceSingleAsync<WaterPoints>()).points;
351
352     points2 = points2 + AppConstants.eightPoints;
353
354     numberOfLogs2 = (await firebaseClient
355     .Child("WaterPoints")
356     .Child(auth.GetUid())
357     .OnceSingleAsync<WaterPoints>()).numberOfLogs;
358
359     numberOfLogs2++;
360
361     cisternCount2 = (await firebaseClient
362     .Child("WaterPoints")
363     .Child(auth.GetUid())
364     .OnceSingleAsync<WaterPoints>()).cisternCount;
365
366     rainBarrel2 = (await firebaseClient
367     .Child("WaterPoints")
368     .Child(auth.GetUid())
369     .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
370
371     reWater2 = (await firebaseClient
372     .Child("WaterPoints")
373     .Child(auth.GetUid())
374     .OnceSingleAsync<WaterPoints>()).reWaterCount;
375
376     showerBucket2 = (await firebaseClient
377     .Child("WaterPoints")
378     .Child(auth.GetUid())
379     .OnceSingleAsync<WaterPoints>()).showerBucketCount;
380
381     showerBucket2++;
382
383     wSShowerHead2 = (await firebaseClient
384     .Child("WaterPoints")
385     .Child(auth.GetUid())
386     .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
387
388     await firebaseClient
389     .Child("WaterPoints")
390     .Child(auth.GetUid())
391     .PutAsync(new WaterPoints()
392     {
393         username = username,
394         points = points2,
395         numberOfLogs = numberOfLogs2,
396         cisternCount = cisternCount2,
397         rainBarrelCount = rainBarrel2,
398         reWaterCount = reWater2,
399         showerBucketCount = showerBucket2,
400         wSShowerHeadCount = wSShowerHead2,
401     });
402 }
403 catch (FirebaseException)
404 {
405     username = (await firebaseClient
406     .Child("users")
407     .Child(auth.GetUid())
408     .OnceSingleAsync<Users>()).username;
409
410     points2 = AppConstants.eightPoints;
```

```
411         await firebaseClient
412             .Child("WaterPoints")
413             .Child(auth.GetUid())
414             .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, showerBucketCount = 1 }); ;
415     }
416 }
417 catch (NullReferenceException)
418 {
419     username = (await firebaseClient
420         .Child("users")
421         .Child(auth.GetUid())
422         .OnceSingleAsync<Users>()).username;
423
424     points2 = AppConstants.eightPoints;
425     await firebaseClient
426         .Child("WaterPoints")
427         .Child(auth.GetUid())
428         .PutAsync(new WaterPoints() { username = username, points = points2,
numberOfLogs = 1, showerBucketCount = 1 }); ;
429     }
430 }
431 /** This function updates the points in the Water category by ten points. It also
increments the number of logs logged in the Water
432 * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
433 */
434 public async void WSSowerHeadPoints()
435 {
436
437     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
438     auth = DependencyService.Get<IAuth>();
439
440     try
441     {
442         username = (await firebaseClient
443             .Child("users")
444             .Child(auth.GetUid())
445             .OnceSingleAsync<Users>()).username;
446
447         points2 = (await firebaseClient
448             .Child("WaterPoints")
449             .Child(auth.GetUid())
450             .OnceSingleAsync<WaterPoints>()).points;
451
452         points2 = points2 + AppConstants.tenPoints;
453
454         numberOfLogs2 = (await firebaseClient
455             .Child("WaterPoints")
456             .Child(auth.GetUid())
457             .OnceSingleAsync<WaterPoints>()).numberOfLogs;
458
459         numberOfLogs2++;
460
461         cisternCount2 = (await firebaseClient
462             .Child("WaterPoints")
463             .Child(auth.GetUid())
464             .OnceSingleAsync<WaterPoints>()).cisternCount;
465
466         rainBarrel2 = (await firebaseClient
467             .Child("WaterPoints")
```

```

468         .Child(auth.GetUid())
469         .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
470
471         reWater2 = (await firebaseClient
472         .Child("WaterPoints")
473         .Child(auth.GetUid())
474         .OnceSingleAsync<WaterPoints>()).reWaterCount;
475
476         showerBucket2 = (await firebaseClient
477         .Child("WaterPoints")
478         .Child(auth.GetUid())
479         .OnceSingleAsync<WaterPoints>()).showerBucketCount;
480
481         wSShowerHead2 = (await firebaseClient
482         .Child("WaterPoints")
483         .Child(auth.GetUid())
484         .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
485
486         wSShowerHead2++;
487
488         await firebaseClient
489         .Child("WaterPoints")
490         .Child(auth.GetUid())
491         .PutAsync(new WaterPoints()
492         {
493             username = username,
494             points = points2,
495             numberOfLogs = numberOfLogs2,
496             cisternCount = cisternCount2,
497             rainBarrelCount = rainBarrel2,
498             reWaterCount = reWater2,
499             showerBucketCount = showerBucket2,
500             wSShowerHeadCount = wSShowerHead2,
501         });
502     }
503     catch (FirebaseException)
504     {
505         username = (await firebaseClient
506         .Child("users")
507         .Child(auth.GetUid())
508         .OnceSingleAsync<Users>()).username;
509
510         points2 = AppConstants.tenPoints;
511         await firebaseClient
512         .Child("WaterPoints")
513         .Child(auth.GetUid())
514         .PutAsync(new WaterPoints() { username = username, points = points2,
515         numberOfLogs = 1, wSShowerHeadCount = 1 }); ;
516     }
517     catch (NullReferenceException)
518     {
519         username = (await firebaseClient
520         .Child("users")
521         .Child(auth.GetUid())
522         .OnceSingleAsync<Users>()).username;
523
524         points2 = AppConstants.tenPoints;
525         await firebaseClient
526         .Child("WaterPoints")
527         .Child(auth.GetUid())

```

```
528 |         .PutAsync(new WaterPoints() { username = username, points = points2,  
529 | numberOfLogs = 1, wSShowerHeadCount = 1 });  
530 |     }  
531 | }  
532 }
```

```
1  /*! \class The WorkPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the WorkPointsUpdate ViewModel Class. It updates the data for the
7  * work by reading in all the chosen data and updating the selected fields and then sending
8  * this data to back firebase.
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class WorkPointsUpdate
19     {
20         int points2 = 0;
21         int numberOfLogs2 = 0;
22         int paperCount2 = 0;
23         int offElectronicsCount2 = 0;
24         int remoteWorkCount2 = 0;
25
26         string username = "";
27
28         IAuth auth;
29         /** This function updates the points in the Work category by four points. It also
30 increments the number of logs logged in the Work
31 * category by one and increments the number of times this particular action was
32 logged by one and sends this data to Firebase.
33 */
34         public async void PaperPoints()
35         {
36             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
37 quake-default-rtdb.firebaseio.com/");
38             auth = DependencyService.Get<IAuth>();
39
40             try
41             {
42                 username = (await firebaseClient
43 .Child("users")
44 .Child(auth.GetUid())
45 .OnceSingleAsync<Users>()).username;
46
47                 points2 = (await firebaseClient
48 .Child("WorkPoints")
49 .Child(auth.GetUid())
50 .OnceSingleAsync<WorkPoints>()).points;
51
52                 points2 = points2 + AppConstants.fourPoints;
53
54                 numberOfLogs2 = (await firebaseClient
55 .Child("WorkPoints")
56 .Child(auth.GetUid())
57 .OnceSingleAsync<WorkPoints>()).numberOfLogs;
```

```
57         numberOfLogs2++;
58
59         paperCount2 = (await firebaseClient
60             .Child("WorkPoints")
61             .Child(auth.GetUid())
62             .OnceSingleAsync<WorkPoints>()).paperCount;
63
64         paperCount2++;
65
66         offElectronicsCount2 = (await firebaseClient
67             .Child("WorkPoints")
68             .Child(auth.GetUid())
69             .OnceSingleAsync<WorkPoints>()).offElectronicsCount;
70
71         remoteWorkCount2 = (await firebaseClient
72             .Child("WorkPoints")
73             .Child(auth.GetUid())
74             .OnceSingleAsync<WorkPoints>()).remoteWorkCount;
75
76         await firebaseClient
77             .Child("WorkPoints")
78             .Child(auth.GetUid())
79             .PutAsync(new WorkPoints()
80             {
81                 username = username,
82                 points = points2,
83                 numberOfLogs = numberOfLogs2,
84                 paperCount = paperCount2,
85                 offElectronicsCount = offElectronicsCount2,
86                 remoteWorkCount = remoteWorkCount2,
87             });
88     }
89     catch (FirebaseException)
90     {
91         username = (await firebaseClient
92             .Child("users")
93             .Child(auth.GetUid())
94             .OnceSingleAsync<Users>()).username;
95
96         points2 = AppConstants.fourPoints;
97         await firebaseClient
98             .Child("WorkPoints")
99             .Child(auth.GetUid())
100             .PutAsync(new WorkPoints() { username = username, points = points2,
101 numberOfLogs = 1, paperCount = 1 }); ;
102     }
103     catch (NullReferenceException)
104     {
105         username = (await firebaseClient
106             .Child("users")
107             .Child(auth.GetUid())
108             .OnceSingleAsync<Users>()).username;
109
110         points2 = AppConstants.fourPoints;
111         await firebaseClient
112             .Child("WorkPoints")
113             .Child(auth.GetUid())
114             .PutAsync(new WorkPoints() { username = username, points = points2,
115 numberOfLogs = 1, paperCount = 1 }); ;
116     }
```

```
117     /** This function updates the points in the Work category by six points. It also
increments the number of logs logged in the Work
118     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
119     */
120     public async void ElectronicsOffPoints()
121     {
122
123         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
124         auth = DependencyService.Get<IAuth>();
125
126         try
127         {
128             username = (await firebaseClient
129                 .Child("users")
130                 .Child(auth.GetUid())
131                 .OnceSingleAsync<Users>()).username;
132
133             points2 = (await firebaseClient
134                 .Child("WorkPoints")
135                 .Child(auth.GetUid())
136                 .OnceSingleAsync<WorkPoints>()).points;
137
138             points2 = points2 + AppConstants.sixPoints;
139
140             numberOfLogs2 = (await firebaseClient
141                 .Child("WorkPoints")
142                 .Child(auth.GetUid())
143                 .OnceSingleAsync<WorkPoints>()).numberOfLogs;
144
145             numberOfLogs2++;
146
147             paperCount2 = (await firebaseClient
148                 .Child("WorkPoints")
149                 .Child(auth.GetUid())
150                 .OnceSingleAsync<WorkPoints>()).paperCount;
151
152             offElectronicsCount2 = (await firebaseClient
153                 .Child("WorkPoints")
154                 .Child(auth.GetUid())
155                 .OnceSingleAsync<WorkPoints>()).offElectronicsCount;
156
157             offElectronicsCount2++;
158
159             remoteWorkCount2 = (await firebaseClient
160                 .Child("WorkPoints")
161                 .Child(auth.GetUid())
162                 .OnceSingleAsync<WorkPoints>()).remoteWorkCount;
163
164             await firebaseClient
165                 .Child("WorkPoints")
166                 .Child(auth.GetUid())
167                 .PutAsync(new WorkPoints()
168                 {
169                     username = username,
170                     points = points2,
171                     numberOfLogs = numberOfLogs2,
172                     paperCount = paperCount2,
173                     offElectronicsCount = offElectronicsCount2,
174                     remoteWorkCount = remoteWorkCount2,
175                 });
```

```

176     }
177     catch (FirebaseException)
178     {
179         username = (await firebaseClient
180             .Child("users")
181             .Child(auth.GetUid())
182             .OnceSingleAsync<Users>()).username;
183
184         points2 = AppConstants.sixPoints;
185         await firebaseClient
186             .Child("WorkPoints")
187             .Child(auth.GetUid())
188             .PutAsync(new WorkPoints() { username = username, points = points2,
189 numberOfLogs = 1, offElectronicsCount = 1 }); ;
190     }
191     catch (NullReferenceException)
192     {
193         username = (await firebaseClient
194             .Child("users")
195             .Child(auth.GetUid())
196             .OnceSingleAsync<Users>()).username;
197
198         points2 = AppConstants.sixPoints;
199         await firebaseClient
200             .Child("WorkPoints")
201             .Child(auth.GetUid())
202             .PutAsync(new WorkPoints() { username = username, points = points2,
203 numberOfLogs = 1, offElectronicsCount = 1 }); ;
204     }
205     /** This function updates the points in the Work category by ten points. It also
206 increments the number of logs logged in the Work
207 * category by one and increments the number of times this particular action was
208 logged by one and sends this data to Firebase.
209 */
210     public async void RemoteWorkPoints()
211     {
212         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
213 quake-default-rtdb.firebaseio.com/");
214         auth = DependencyService.Get<IAuth>();
215
216         try
217         {
218             username = (await firebaseClient
219                 .Child("users")
220                 .Child(auth.GetUid())
221                 .OnceSingleAsync<Users>()).username;
222
223             points2 = (await firebaseClient
224                 .Child("WorkPoints")
225                 .Child(auth.GetUid())
226                 .OnceSingleAsync<WorkPoints>()).points;
227
228             points2 = points2 + AppConstants.tenPoints;
229
230             numberOfLogs2 = (await firebaseClient
231                 .Child("WorkPoints")
232                 .Child(auth.GetUid())
233                 .OnceSingleAsync<WorkPoints>()).numberOfLogs;

```



```
233         numberOfLogs2++;
234
235         paperCount2 = (await firebaseClient
236             .Child("WorkPoints")
237             .Child(auth.GetUid())
238             .OnceSingleAsync<WorkPoints>()).paperCount;
239
240         offElectronicsCount2 = (await firebaseClient
241             .Child("WorkPoints")
242             .Child(auth.GetUid())
243             .OnceSingleAsync<WorkPoints>()).offElectronicsCount;
244
245         remoteWorkCount2 = (await firebaseClient
246             .Child("WorkPoints")
247             .Child(auth.GetUid())
248             .OnceSingleAsync<WorkPoints>()).remoteWorkCount;
249
250         remoteWorkCount2++;
251
252         await firebaseClient
253             .Child("WorkPoints")
254             .Child(auth.GetUid())
255             .PutAsync(new WorkPoints()
256                 {
257                     username = username,
258                     points = points2,
259                     numberOfLogs = numberOfLogs2,
260                     paperCount = paperCount2,
261                     offElectronicsCount = offElectronicsCount2,
262                     remoteWorkCount = remoteWorkCount2,
263                 });
264     }
265     catch (FirebaseException)
266     {
267         username = (await firebaseClient
268             .Child("users")
269             .Child(auth.GetUid())
270             .OnceSingleAsync<Users>()).username;
271
272         points2 = AppConstants.tenPoints;
273         await firebaseClient
274             .Child("WorkPoints")
275             .Child(auth.GetUid())
276             .PutAsync(new WorkPoints() { username = username, points = points2,
277 numberOfLogs = 1, remoteWorkCount = 1 }); ;
278     }
279     catch (NullReferenceException)
280     {
281         username = (await firebaseClient
282             .Child("users")
283             .Child(auth.GetUid())
284             .OnceSingleAsync<Users>()).username;
285
286         points2 = AppConstants.tenPoints;
287         await firebaseClient
288             .Child("WorkPoints")
289             .Child(auth.GetUid())
290             .PutAsync(new WorkPoints() { username = username, points = points2,
291 numberOfLogs = 1, remoteWorkCount = 1 }); ;
292     }
    }
```

```
293 | }  
294 | }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     xmlns:local="clr-namespace:Application_Green_Quake.Models"
5     x:Class="Application_Green_Quake.Views.SignUpPage"
6     NavigationPage.HasNavigationBar="False">
7     <ContentPage.Content>
8         <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
9             <Grid.RowDefinitions>
10                <RowDefinition Height="2*" />
11                <RowDefinition Height="2*" />
12                <RowDefinition Height="*" />
13            </Grid.RowDefinitions>
14
15            <Image Grid.Row="0"
16                Source="{local:ImageResource Application_Green_Quake.Images.Logo.PNG}"
17                Aspect="AspectFit" />
18
19            <StackLayout Grid.Row="1" BackgroundColor="White" Padding="0,10,0,0">
20                <Entry Placeholder="Username"
21                    x:Name="UsernameInput"
22                    HorizontalTextAlignment="Center"
23                    PlaceholderColor="White"
24                    TextColor="Black"
25                    BackgroundColor="#50C878"
26                    Margin="60,0,60,0" />
27                <Label x:Name="UsernameErrorLabel"
28                    TextColor="Red"
29                    HorizontalTextAlignment="Center" />
30                <Entry Placeholder="Email"
31                    Keyboard="Email"
32                    x:Name="EmailInput"
33                    HorizontalTextAlignment="Center"
34                    PlaceholderColor="White"
35                    TextColor="Black"
36                    BackgroundColor="#50C878"
37                    Margin="60,0,60,0" />
38                <Label x:Name="EmailErrorLabel"
39                    TextColor="Red"
40                    HorizontalTextAlignment="Center" />
41                <Entry Grid.Row="1"
42                    Placeholder="Password"
43                    HorizontalTextAlignment="Center"
44                    IsPassword="true"
45                    x:Name="PasswordInput"
46                    PlaceholderColor="White"
47                    TextColor="Black"
48                    BackgroundColor="#50C878"
49                    Margin="60,0,60,0" />
50                <Label x:Name="PasswordErrorLabel"
51                    Text="Password must be at least 8 chars long and contain an upper
52                    case letter, lower case letter and a number."
53                    TextColor="Black"
54                    HorizontalTextAlignment="Center" />
55            </StackLayout>
56            <StackLayout Grid.Row="2" BackgroundColor="White" VerticalOptions="Center">
57                <Button Text="Sign Up"
58                    Clicked="SignUpClicked"
59                    CornerRadius="30"
60                    BackgroundColor="#50C878"
61                    TextColor="White" />
62            </StackLayout>
63        </Grid>
64    </ContentPage.Content>
65 </ContentPage>
```

```
61         Margin="60,0,60,0"/>
62     </StackLayout>
63 </Grid>
64
65 </ContentPage.Content>
66 </ContentPage>
```

```

1  /!* \class The SignUpPage View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * 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.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Firebase.Database;
11 using Firebase.Database.Query;
12 using System;
13 using System.Globalization;
14 using System.Text.RegularExpressions;
15 using Acr.UserDialogs;
16 using Xamarin.Forms;
17 using Xamarin.Forms.Xaml;
18
19 namespace Application_Green_Quake.Views
20 {
21     [XamlCompilation(XamlCompilationOptions.Compile)]
22     public partial class SignUpPage : ContentPage
23     {
24         IAuth auth;
25         public SignUpPage()
26         {
27             InitializeComponent();
28             auth = DependencyService.Get<IAuth>();
29         }
30         /** 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
31         * the correct error message is displayed.
32         */
33         async void SignUpClicked(object sender, EventArgs e)
34         {
35             var emailPattern = @"^(?(\")(\\".+?(?<!\\\\)\"@)|((([0-9a-z](\\.(?!\\.)))|
   [-!#\$\%&'\\*\+\/=\\?\^\`\\{\}\|\~\w]*)?(?<=[0-9a-z])@)(?(\[)(\\[\\d{1,3}\\.
   {3}\\d{1,3}\\.\\d{1,3}\\.\\d{1,3})|((([0-9a-z]|\w)*[0-9a-z]*\w)+[a-z0-9][\w-a-z0-9]{0,22}[a-z0-9]))$";
36             var hasNum = new Regex(@"[0-9]+");
37             var hasUpperChar = new Regex(@"[A-Z]+");
38             var hasLowerChar = new Regex(@"[a-z]+");
39             var hasSpecialChar = new Regex(@"[^\\w]+");
40             var hasMinimum8Chars = new Regex(@".{8,}");
41
42             UsernameErrorLabel.Text = null;
43             EmailErrorLabel.Text = null;
44             PasswordErrorLabel.Text = null;
45
46             if (UsernameInput.Text == null && EmailInput.Text != null && PasswordInput.Text
   != null)
47             {
48
49                 UsernameInput.Text = null;
50                 UsernameErrorLabel.Text = "No Username Entered";
51             }
52             else if (UsernameInput.Text != null && EmailInput.Text == null &&
   PasswordInput.Text != null)
53             {
54                 EmailInput.Text = null;
55                 EmailErrorLabel.Text = "No Email Entered";

```

```
56     }
57     else if (UsernameInput.Text != null && EmailInput.Text != null &&
PasswordInput.Text == null)
58     {
59
60         PasswordInput.Text = null;
61         PasswordErrorLabel.TextColor = Color.Red;
62         PasswordErrorLabel.Text = "No Password Entered";
63     }
64     else if (UsernameInput.Text != null && EmailInput.Text == null &&
PasswordInput.Text == null)
65     {
66         EmailInput.Text = null;
67         PasswordInput.Text = null;
68         PasswordErrorLabel.TextColor = Color.Red;
69         PasswordErrorLabel.Text = "No Password Entered";
70         EmailErrorLabel.Text = "No Email Entered";
71     }
72     else if (UsernameInput.Text == null && EmailInput.Text != null &&
PasswordInput.Text == null)
73     {
74         PasswordInput.Text = null;
75         UsernameInput.Text = null;
76         PasswordErrorLabel.TextColor = Color.Red;
77         UsernameErrorLabel.Text = "No Username Entered";
78         PasswordErrorLabel.Text = "No Password Entered";
79         await DisplayAlert("Sign Up Failed", "No Username or Password", "Ok");
80     }
81     else if (UsernameInput.Text == null && EmailInput.Text == null &&
PasswordInput.Text != null)
82     {
83         EmailInput.Text = null;
84
85         UsernameInput.Text = null;
86         UsernameErrorLabel.Text = "No Username Entered";
87         EmailErrorLabel.Text = "No Email Entered";
88     }
89     else if (UsernameInput.Text == null && EmailInput.Text == null &&
PasswordInput.Text == null)
90     {
91         EmailInput.Text = null;
92         PasswordInput.Text = null;
93         UsernameInput.Text = null;
94         PasswordErrorLabel.TextColor = Color.Red;
95         EmailErrorLabel.Text = "No Email Entered";
96         PasswordErrorLabel.Text = "No Password Entered";
97         UsernameErrorLabel.Text = "No Username Entered";
98     }
99
100     if (UsernameInput.Text != null && EmailInput.Text != null && PasswordInput.Text
!= null)
101     {
102         if (!Regex.IsMatch(EmailInput.Text, emailPattern))
103         {
104             EmailInput.Text = null;
105             EmailErrorLabel.Text = "Email is invalid";
106         }
107         if (!hasNum.IsMatch(PasswordInput.Text))
108         {
109             PasswordInput.Text = null;
110             PasswordErrorLabel.TextColor = Color.Red;
111             PasswordErrorLabel.Text = "Password must have at least one number";
```

```

112     }
113     else if (!hasLowerChar.IsMatch(PasswordInput.Text))
114     {
115         PasswordInput.Text = null;
116         PasswordErrorLabel.TextColor = Color.Red;
117         PasswordErrorLabel.Text = "Password must have at least one lower case
character";
118     }
119     else if (!hasUpperChar.IsMatch(PasswordInput.Text))
120     {
121         PasswordInput.Text = null;
122         PasswordErrorLabel.TextColor = Color.Red;
123         PasswordErrorLabel.Text = "Password must have at least one upper case
character";
124     }
125     else if (!hasSpecialChar.IsMatch(PasswordInput.Text))
126     {
127         PasswordInput.Text = null;
128         PasswordErrorLabel.TextColor = Color.Red;
129         PasswordErrorLabel.Text = "Password must have at least one special
character";
130     }
131     else if (!hasMinimum8Chars.IsMatch(PasswordInput.Text))
132     {
133         PasswordInput.Text = null;
134         PasswordErrorLabel.TextColor = Color.Red;
135         PasswordErrorLabel.Text = "Password must be at least 8 characters";
136     }
137     else if (EmailErrorLabel.Text == null && PasswordErrorLabel.Text == null &&
UsernameErrorLabel.Text == null)
138     {
139
140         var user = auth.SignUpWithEmailAndPassword(EmailInput.Text,
PasswordInput.Text);
141         if (user != null)
142         {
143             var signOut = auth.SignOut();
144
145             FirebaseClient firebaseClient = new
FirebaseClient("https://application-green-quake-default-rtdb.firebaseio.com/");
146
147             string usernameInput = UsernameInput.Text;
148             string token = await user;
149             string theBio = "";
150             string theNation = RegionInfo.CurrentRegion.EnglishName;
151
152             if (token != "duplicate")
153             {
154                 UserDialogs.Instance.ShowLoading("");
155
156                 await firebaseClient
157                     .Child("users")
158                     .Child(token)
159                     .PutAsync(new Users() { username = usernameInput, bio =
theBio, nation = theNation});
160
161                 await firebaseClient
162                     .Child("usernames")
163                     .Child(usernameInput)
164                     .PutAsync(new Usernames() { Uid = token });
165
166                 if (signOut)

```

```
167         {
168             UserDialogs.Instance.HideLoading();
169             await DisplayAlert("Success", "New User Created", "OK");
170             await Navigation.PushAsync(new MainPage());
171         }
172         else
173         {
174             await DisplayAlert("Error", "An error has occurred, please
try again", "Ok");
175         }
176     }
177     else
178     {
179         await DisplayAlert("Error", "The email already exists, please
try again.", "Ok");
180     }
181 }
182 else
183 {
184     await DisplayAlert("Error", "Please connect to the internet.",
"Ok");
185 }
186 }
187 }
188 }
189 }
190 }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3   xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4   xmlns:local="clr-namespace:Application_Green_Quake.Models"
5   x:Class="Application_Green_Quake.Views.MainPage"
6   NavigationPage.HasNavigationBar="False">
7   <ContentPage.Content>
8     <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
9       <Grid.RowDefinitions>
10        <RowDefinition Height="2*"/>
11        <RowDefinition Height="*"/>
12        <RowDefinition Height="*"/>
13      </Grid.RowDefinitions>
14
15      <Image
16        Grid.Row="0"
17        Source="{local:ImageResource Application_Green_Quake.Images.Logo.PNG}"
18        Aspect="AspectFit"/>
19
20      <StackLayout Grid.Row="1" BackgroundColor="White">
21        <Entry Placeholder="Email"
22          Keyboard="Email"
23          x:Name="EmailInput"
24          HorizontalTextAlignment="Center"
25          PlaceholderColor="White"
26          TextColor="Black"
27          BackgroundColor="#50C878"
28          Margin="60,0,60,0"/>
29        <Label x:Name="EmailErrorLabel"
30          TextColor="Red"
31          HorizontalTextAlignment="Center"/>
32        <Entry Grid.Row="1"
33          Placeholder="Password"
34          HorizontalTextAlignment="Center"
35          IsPassword="true"
36          x:Name="PasswordInput"
37          PlaceholderColor="White"
38          TextColor="Black"
39          BackgroundColor="#50C878"
40          Margin="60,0,60,0"/>
41        <Label x:Name="PasswordErrorLabel"
42          TextColor="Red"
43          HorizontalTextAlignment="Center"/>
44      </StackLayout>
45      <StackLayout Grid.Row="2" BackgroundColor="White">
46        <Button Text="Login"
47          Clicked="LoginClicked"
48          CornerRadius="30"
49          BackgroundColor="#50C878"
50          TextColor="White"
51          Margin="60,0,60,0"/>
52        <Button Text="Sign Up"
53          TextColor="#50C878"
54          HorizontalOptions="Center"
55          BackgroundColor="Transparent"
56          Clicked="SignUpClicked" />
57        <Button Text="Forgot Password?"
58          HorizontalOptions="Center"
59          BackgroundColor="Transparent"
60          Clicked="ForgotPasswordClicked"
61          Padding="0,0,0,30"/>
```

```
62 |                                     TextColor="Black"/>
63 |                                     </StackLayout>
64 |                                 </Grid>
65 | </ContentPage.Content>
66 </ContentPage>
```

```
1 /!* \class The MainPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the MainPage View Class. This is the class that allows a user to
   login to the application. It contains validation checks.
7 *
8 */
9 using System;
10 using Application_Green_Quake.ViewModels;
11 using Xamarin.Forms;
12 using Xamarin.Forms.Xaml;
13
14 namespace Application_Green_Quake.Views
15 {
16     [XamlCompilation(XamlCompilationOptions.Compile)]
17     public partial class MainPage : ContentPage
18     {
19         IAuth auth;
20         public static string token;
21         public MainPage()
22         {
23             InitializeComponent();
24             auth = DependencyService.Get<IAuth>();
25         }
26         /** 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
27         * the correct error message is displayed.
28         */
29         async void LoginClicked(object sender, EventArgs e)
30         {
31             EmailErrorLabel.Text = null;
32             PasswordErrorLabel.Text = null;
33
34             if (EmailInput.Text == null && PasswordInput.Text != null)
35             {
36                 EmailInput.Text = null;
37                 PasswordInput.Text = null;
38                 EmailErrorLabel.Text = "No Email Entered";
39             }
40             else if (PasswordInput.Text == null && EmailInput.Text != null)
41             {
42                 EmailInput.Text = null;
43                 PasswordInput.Text = null;
44                 PasswordErrorLabel.Text = "No Password Entered";
45             }
46             else if (EmailInput.Text == null && PasswordInput.Text == null)
47             {
48                 EmailInput.Text = null;
49                 PasswordInput.Text = null;
50                 EmailErrorLabel.Text = "No Email Entered";
51                 PasswordErrorLabel.Text = "No Password Entered";
52             }
53             else
54             {
55                 try
56                 {
57                     token = await auth.LoginWithEmailAndPassword(EmailInput.Text,
   PasswordInput.Text);
```

```
58         if (token != string.Empty)
59         {
60             GetData level = new GetData();
61             level.SetLvl();
62             await Navigation.PushAsync(new LoginSplashPage());
63         }
64         else
65         {
66             EmailInput.Text = null;
67             PasswordInput.Text = null;
68             await DisplayAlert("Authentication Failed", "Email or Password are
incorrect", "Ok");
69         }
70     }
71     catch (Exception)
72     {
73         await DisplayAlert("Authentication Failed", "Please connect to the
internet", "Ok");
74     }
75 }
76 }
77 }
78 /** 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.
79 */
80 void SignUpClicked(object sender, EventArgs e)
81 {
82     var signOut = auth.SignOut();
83
84     if (signOut)
85     {
86         Navigation.PushAsync(new SignUpPage());
87     }
88 }
89 /** This function fires when the Forgot Password Text is clicked. It redirects to
the Forgot Password page.
90 */
91 private async void ForgotPasswordClicked(object sender, EventArgs e)
92 {
93     await Navigation.PushAsync(new ForgotPasswordPage());
94 }
95 }
96 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2
3 <TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
4     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:views="clr-
namespace:Application_Green_Quake.Views"
5     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"
6     x:Class="Application_Green_Quake.Views.MainMenu"
7     NavigationPage.HasBackButton="False"
8     xmlns:android="clr-
namespace:Xamarin.Forms.PlatformConfiguration.AndroidSpecific;assembly=Xamarin.Forms.Core"
9     android:TabbedPage.ToolbarPlacement="Bottom"
10     UnselectedTabColor="Black"
11     SelectedTabColor="White" >
12
13     <NavigationPage.TitleView>
14         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
15             <Label Text="Green Quake" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
16             <Label x:Name="theLevel" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
Margin="0,0,30,0"/>
17         </StackLayout>
18     </NavigationPage.TitleView>
19
20     <ContentPage Title="Home" IconImageSource="{local:ImageResource
Application_Green_Quake.Images.Tabbed.Home.png}">
21         <ScrollView>
22             <Grid VerticalOptions="FillAndExpand" RowSpacing="5">
23                 <Grid.RowDefinitions>
24                     <RowDefinition Height="2*"/>
25                     <RowDefinition Height="2*"/>
26                     <RowDefinition Height="*/>
27                 </Grid.RowDefinitions>
28
29                 <Image Grid.Column="0"
30                     Grid.Row="0"
31                     Aspect="AspectFill"
32                     Source="{local:ImageResource
Application_Green_Quake.Images.EcoActions.jpg}"/>
33                 <StackLayout Grid.Column="0"
34                     Grid.Row="0"
35                     BackgroundColor="Black"
36                     Opacity=".4">
37                     <StackLayout.GestureRecognizers>
38                         <TapGestureRecognizer Tapped="NavigateToEcoActionButton"/>
39                     </StackLayout.GestureRecognizers>
40                 </StackLayout>
41                 <StackLayout Grid.Column="0"
42                     Grid.Row="0"
43                     VerticalOptions="Center"
44                     Spacing="5"
45                     Margin="15,0,40,15">
46                     <StackLayout.GestureRecognizers>
47                         <TapGestureRecognizer Tapped="NavigateToEcoActionButton"/>
48                     </StackLayout.GestureRecognizers>
49                     <Label Text="Eco Actions"
50                         TextColor="White"
51                         FontSize="24"

```

```
52         FontAttributes="Bold"
53         FontFamily="Proxima Nova"/>
54         <Label Text="Log your Eco Friendly Activities and earn rewards!"
55             TextColor="White"
56             FontSize="15"
57             FontFamily="Proxima Nova Thin"/>
58     </StackLayout>
59
60     <Image Grid.Column="0"
61         Grid.Row="1"
62         Aspect="AspectFill"
63         Source="{local:ImageResource
Application_Green_Quake.Images.Refill.jpg}"/>
64     <StackLayout Grid.Column="0"
65         Grid.Row="1"
66         BackgroundColor="Black"
67         Opacity=".4">
68         <StackLayout.GestureRecognizers>
69             <TapGestureRecognizer Tapped="NavigateToRefillStation"/>
70         </StackLayout.GestureRecognizers>
71     </StackLayout>
72     <StackLayout Grid.Column="0"
73         Grid.Row="1"
74         VerticalOptions="Center"
75         Spacing="5"
76         Margin="15,0,40,15">
77         <StackLayout.GestureRecognizers>
78             <TapGestureRecognizer Tapped="NavigateToRefillStation"/>
79         </StackLayout.GestureRecognizers>
80         <Label Text="Refill Stations"
81             TextColor="White"
82             FontSize="24"
83             FontAttributes="Bold"
84             FontFamily="Proxima Nova"/>
85         <Label Text="Find the closest place where you can get a refill. Use
reusable bottles and reduce plastic waste!"
86             TextColor="White"
87             FontSize="15"
88             FontFamily="Proxima Nova Thin" />
89     </StackLayout>
90
91     <Image Grid.Column="0"
92         Grid.Row="2"
93         Aspect="AspectFill"
94         Source="{local:ImageResource
Application_Green_Quake.Images.SignOut.jpg}"/>
95     <StackLayout Grid.Column="0"
96         Grid.Row="2"
97         BackgroundColor="Black"
98         Opacity=".4">
99         <StackLayout.GestureRecognizers>
100             <TapGestureRecognizer Tapped="SignOutButton"/>
101         </StackLayout.GestureRecognizers>
102     </StackLayout>
103     <StackLayout Grid.Column="0"
104         Grid.Row="2"
105         VerticalOptions="Center"
106         Spacing="5"
107         Margin="15,0,40,15">
108         <StackLayout.GestureRecognizers>
109             <TapGestureRecognizer Tapped="SignOutButton"/>
110     </StackLayout.GestureRecognizers>
```

```
111         <Label Text="Sign Out"
112             TextColor="White"
113             FontSize="24"
114             FontAttributes="Bold"
115             FontFamily="Proxima Nova"/>
116     </StackLayout>
117 </Grid>
118 </ScrollView>
119 </ContentPage>
120
121     <views2:TopTabLeaderBoard Title="Leaderboard" IconImageSource="{local:ImageResource
Application_Green_Quake.Images.Tabbed.Leaderboard.png}"></views2:TopTabLeaderBoard>
122     <views1:TopTabProfile Title="Profile" IconImageSource="{local:ImageResource
Application_Green_Quake.Images.Tabbed.Profile.png}"></views1:TopTabProfile>
123 </TabbedPage>
```

```
1 /!* \class The MainMenu View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the MainMenu View Class. This page is the main menu of the
   application and provides navigation to all the apps screens.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.EcoActionsMenu;
11 using Application_Green_Quake.Views.RefillPage;
12 using System;
13 using System.Threading.Tasks;
14 using Acr.UserDialogs;
15 using Xamarin.Forms;
16 using Xamarin.Forms.Xaml;
17 using Application = Xamarin.Forms.Application;
18
19 namespace Application_Green_Quake.Views
20 {
21     [XamlCompilation(XamlCompilationOptions.Compile)]
22     public partial class MainMenu : TabbedPage
23     {
24         IAuth auth;
25         public MainMenu()
26         {
27             InitializeComponent();
28             auth = DependencyService.Get<IAuth>();
29             OnAppearing();
30         }
31
32         /** The constructor for Main menu
33         @param tab supplied to tell the class which tabbed page to display.
34         */
35         public MainMenu(int tab)
36         {
37             InitializeComponent();
38             auth = DependencyService.Get<IAuth>();
39             CurrentPage = Children[tab];
40             OnAppearing();
41         }
42
43         /** This function navigates to ActionCategories
44         */
45         private async void NavigateToEcoActionButton(object sender, EventArgs e)
46         {
47             await Navigation.PushAsync(new ActionsCategories());
48         }
49         /** This function navigates to RefillStation
50         */
51         private async void NavigateToRefillStation(object sender, EventArgs e)
52         {
53             await Navigation.PushAsync(new RefillStation());
54         }
55
56         /** This function Signs the user out and navigates to MainPage
57         */
58         void SignOutButton(object sender, EventArgs e)
59         {
```



```
60     var signOut = auth.SignOut();
61
62     if (signOut)
63     {
64         Application.Current.MainPage = new NavigationPage(new MainPage());
65     }
66 }
67 /** This function is called before the page is displayed.
68 */
69 protected override async void OnAppearing()
70 {
71     // Wait 2 seconds to allow the data to load.
72     await Task.Delay(2000);
73     //Call functions to load and set data.
74     GetData data = new GetData();
75     data.SetLvl();
76     UserDialogs.Instance.HideLoading();
77
78     GetBadgeData badgeData = new GetBadgeData();
79     badgeData.SetBadgeData();
80
81     GetAchievementsData achievementsData = new GetAchievementsData();
82     achievementsData.SetAchievementsData();
83
84     //Set the level in the navigation bar.
85     theLevel.Text = "LVL: " + GetData.lvl;
86
87 }
88 }
89 }
```

```
1 /!* \class The LoginSplashPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the LoginSplashPage View Class. This class provides the splash
   screen when a user logs into the applciation.
7 *
8 */
9 using Xamarin.Forms;
10
11 namespace Application_Green_Quake.Views
12 {
13     public class LoginSplashPage : ContentPage
14     {
15         Image splashImage;
16
17         /** This function fires when the Login button is clicked. It provides the splash
   screen and then navigates to the main menu right after.
18         */
19         public LoginSplashPage()
20         {
21             NavigationPage.SetHasNavigationBar(this, false);
22
23             //Set the image for the splash screen
24             var sub = new AbsoluteLayout();
25             splashImage = new Image
26             {
27                 Source =
   ImageSource.FromResource("Application_Green_Quake.Images.trees.png"),
28                 WidthRequest = 150,
29                 HeightRequest = 150
30             };
31
32             AbsoluteLayout.SetLayoutFlags(splashImage,
33                 AbsoluteLayoutFlags.PositionProportional);
34             AbsoluteLayout.SetLayoutBounds(splashImage,
35                 new Rectangle(0.5, 0.5, AbsoluteLayout.AutoSize, AbsoluteLayout.AutoSize));
36
37             sub.Children.Add(splashImage);
38
39             this.BackgroundColor = Color.FromHex("#50C878");
40             this.Content = sub;
41         }
42
43         protected override async void OnAppearing()
44         {
45             base.OnAppearing();
46             //Set the animation
47             await splashImage.ScaleTo(0.4, 1100, Easing.Linear);
48             await splashImage.ScaleTo(700, 900, Easing.Linear);
49             Application.Current.MainPage = new NavigationPage(new MainMenu());
50         }
51     }
52 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.ForgotPasswordPage"
5     NavigationPage.HasNavigationBar="False">
6     <ContentPage.Content>
7         <StackLayout VerticalOptions="CenterAndExpand" Spacing="20">
8             <Label Text="Forgot Your Password"
9                 HorizontalOptions="CenterAndExpand"
10                TextColor="Black"
11                FontSize="25"
12                FontAttributes="Bold"/>
13             <Entry Placeholder="Email"
14                 Keyboard="Email"
15                 x:Name="EmailInput"
16                 HorizontalTextAlignment="Center"
17                 PlaceholderColor="White"
18                 TextColor="Black"
19                 BackgroundColor="#50C878"
20                 Margin="60,0,60,0"/>
21             <Label x:Name="EmailErrorLabel"
22                 TextColor="Red"
23                 HorizontalTextAlignment="Center"/>
24             <Button Text="Send"
25                 Clicked="OnResetPassword"
26                 CornerRadius="30"
27                 BackgroundColor="#50C878"
28                 TextColor="White"
29                 Margin="60,0,60,0"/>
30         </StackLayout>
31     </ContentPage.Content>
32 </ContentPage>
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.ForgotPasswordPage"
5     NavigationPage.HasNavigationBar="False">
6     <ContentPage.Content>
7         <StackLayout VerticalOptions="CenterAndExpand" Spacing="20">
8             <Label Text="Forgot Your Password"
9                 HorizontalOptions="CenterAndExpand"
10                TextColor="Black"
11                FontSize="25"
12                FontAttributes="Bold"/>
13             <Entry Placeholder="Email"
14                 Keyboard="Email"
15                 x:Name="EmailInput"
16                 HorizontalTextAlignment="Center"
17                 PlaceholderColor="White"
18                 TextColor="Black"
19                 BackgroundColor="#50C878"
20                 Margin="60,0,60,0"/>
21             <Label x:Name="EmailErrorLabel"
22                 TextColor="Red"
23                 HorizontalTextAlignment="Center"/>
24             <Button Text="Send"
25                 Clicked="OnResetPassword"
26                 CornerRadius="30"
27                 BackgroundColor="#50C878"
28                 TextColor="White"
29                 Margin="60,0,60,0"/>
30         </StackLayout>
31     </ContentPage.Content>
32 </ContentPage>
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             xmlns:maps="clr-
namespace:Xamarin.Forms.GoogleMaps;assembly=Xamarin.Forms.GoogleMaps"
5             x:Class="Application_Green_Quake.Views.RefillPage.RefillStation"
6             NavigationPage.HasNavigationBar="False">
7     <ContentPage.Content>
8         <maps:Map x:Name ="map"/>
9     </ContentPage.Content>
10 </ContentPage>
```

```
1 /!* \class The RefillStation View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * 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.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Firebase.Database;
11 using System;
12 using System.Diagnostics;
13 using System.Linq;
14 using Xamarin.Essentials;
15 using Xamarin.Forms;
16 using Xamarin.Forms.GoogleMaps;
17 using Xamarin.Forms.Xaml;
18 namespace Application_Green_Quake.Views.RefillPage
19 {
20     [XamlCompilation(XamlCompilationOptions.Compile)]
21     public partial class RefillStation : ContentPage
22     {
23         public RefillStation()
24         {
25             InitializeComponent();
26             OnAppearing();
27         }
28         /** This function is called before the page is displayed.
29         */
30         protected override async void OnAppearing()
31         {
32             try
33             {
34                 var location = await Geolocation.GetLastKnownLocationAsync();
35                 if (location == null)
36                 {
37                     location = await Geolocation.GetLocationAsync(new GeolocationRequest
38                     {
39                         DesiredAccuracy = GeolocationAccuracy.Medium,
40                         Timeout = TimeSpan.FromSeconds(30)
41                     });
42                 }
43             }
44             else
45             {
46                 Pin currentLocation = new Pin()
47                 {
48                     Type = PinType.SavedPin,
49                     Label = "Me",
50                     Address = "Here",
51                     Position = new Position(location.Latitude, location.Longitude),
52                     Tag = "id_Me",
53                 };
54             }
55             // Add the pin and load the map at this location
56             map.Pins.Add(currentLocation);
57             map.MoveToRegion(MapSpan.FromCenterAndRadius(currentLocation.Position,
58                 Distance.FromMeters(5000)));
59         }
60     }
61 }
```

```
59     }
60 }
61
62 }
63 catch (Exception e)
64 {
65     Debug.WriteLine($"Something is wrong: {e.Message}");
66 }
67
68
69     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
70
71     //Load the pin data into a list
72     var list = (await firebaseClient
73 .Child("Stations")
74 .OnceAsync<Station>()).Select(item => new Station
75 {
76     description = item.Object.description,
77     label = item.Object.label,
78     latitude = item.Object.latitude,
79     longitude = item.Object.longitude,
80
81 }).ToList();
82
83     // For each entry in the data create and place a pin.
84     foreach (var obj in list)
85     {
86         Pin stationLocations = new Pin()
87         {
88             Type = PinType.SavedPin,
89             Label = obj.label,
90             Address = obj.description,
91             Position = new Position(obj.latitude, obj.longitude),
92
93         };
94         map.Pins.Add(stationLocations);
95     }
96 }
97 }
98 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Home.AirOutHome"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Air Out Home" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Home.AirOut.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Air out your home and get rid of those indoor pollutants."
37                         TextColor="Black"
38                         HorizontalTextAlignment="Center"
39                         FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="2 POINTS!"
44                         TextColor="Black"
45                         FontAttributes="Bold"
46                         FontSize="20"
47                         HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49                         BackgroundColor="#50C878"
50                         TextColor="White"
51                         Margin="60,0,60,0"
52                         Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```



```

1  /!* \class The AirOutHome View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the AirOutHome View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class AirOutHome : ContentPage
20     {
21         public AirOutHome()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTwoPoints();
52                 HomePointsUpdate helper2 = new HomePointsUpdate();
53                 helper2.AirOutPoints();
54                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Waste.BillsOnline"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Online Bills" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Waste.OnlineBills.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Pay your bills online. There is no need to get letters
37 anymore. This removes the need to produce more paper and print letters an in turn reduces
38 paper usage and waste."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="4 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```
1 /!* \class The BillsOnline View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the BillsOnline View Class. This class is the eco action that the
   user can log.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class BillsOnline : ContentPage
20     {
21         public BillsOnline()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 WastePointsUpdate helper2 = new WastePointsUpdate();
53                 helper2.BillsPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }
```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Work.BothSidesPaper"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Both Sides" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Work.Paper.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Use both sides of the paper. This reduces paper waste and
36 the amount of paper needed."
37                         TextColor="Black"
38                         HorizontalTextAlignment="Center"
39                         FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="4 POINTS!"
44                         TextColor="Black"
45                         FontAttributes="Bold"
46                         FontSize="20"
47                         HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49                         BackgroundColor="#50C878"
50                         TextColor="White"
51                         Margin="60,0,60,0"
52                         Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```

```
1 using Application_Green_Quake.Models;
2 using Application_Green_Quake.ViewModels;
3 using System;
4 using System.Threading.Tasks;
5 using Xamarin.Forms;
6 using Xamarin.Forms.Xaml;
7
8 namespace Application_Green_Quake.Views.EcoActions.Work
9 {
10     [XamlCompilation(XamlCompilationOptions.Compile)]
11     public partial class BothSidesPaper : ContentPage
12     {
13         public BothSidesPaper()
14         {
15             InitializeComponent();
16             OnAppearing();
17         }
18         /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
19         * methods
20         */
21         private async void AddPointsClicked(object sender, EventArgs e)
22         {
23             SecurityMethods checks = new SecurityMethods();
24             Task<bool> myTask = checks.DayLimitLock();
25             await myTask;
26
27             Task<bool> myTaskTwo = checks.TimeLimitLock();
28             await myTaskTwo;
29
30             if (myTask.Result)
31             {
32                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
33                 await Navigation.PushAsync(new MainMenu());
34             }
35             else if (myTaskTwo.Result)
36             {
37                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
38                 await Navigation.PushAsync(new MainMenu());
39             }
40             else
41             {
42                 PointsUpdate helper = new PointsUpdate();
43                 helper.UpdateByFourPoints();
44                 WorkPointsUpdate helper2 = new WorkPointsUpdate();
45                 helper2.PaperPoints();
46                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
47                 await Navigation.PushAsync(new MainMenu());
48             }
49         }
50         /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
51         * methods
52         */
53         protected override void OnAppearing()
54         {
55             GetData data = new GetData();
56             data.SetLvl();
57         }
58     }
59 }
```

```
58 |         theLevel.Text = "LVL: " + GetData.lvl.ToString();  
59 |     }  
60 | }  
61 | }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Habits.BrushingTeeth"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Tap Off" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*" />
22                     <RowDefinition Height="2*" />
23                     <RowDefinition Height="*" />
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Habits.TimedBrushing.jpg}" />
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="When brushing your teeth turn of the water. This helps save
36 water while water waste is becoming a major problem in todays world. Water shortages and
37 draught are increasing throughout the globe."
38                         TextColor="Black"
39                         HorizontalTextAlignment="Center"
40                         FontSize="20" />
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="2 POINTS!"
45                         TextColor="Black"
46                         FontAttributes="Bold"
47                         FontSize="20"
48                         HorizontalOptions="Center" />
49                     <Button Text="Completed"
50                         BackgroundColor="#50C878"
51                         TextColor="White"
52                         Margin="60,0,60,0"
53                         Clicked="AddPointsClicked" />
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The BrushingTeeth View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the BrushingTeeth View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13
14 using Xamarin.Forms;
15 using Xamarin.Forms.Xaml;
16
17 namespace Application_Green_Quake.Views.EcoActions.Habits
18 {
19     [XamlCompilation(XamlCompilationOptions.Compile)]
20     public partial class BrushingTeeth : ContentPage
21     {
22         public BrushingTeeth()
23         {
24             InitializeComponent();
25             OnAppearing();
26         }
27         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
42                 await Navigation.PushAsync(new MainMenu());
43             }
44             else if (myTaskTwo.Result)
45             {
46                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
47                 await Navigation.PushAsync(new MainMenu());
48             }
49             else
50             {
51                 PointsUpdate helper = new PointsUpdate();
52                 helper.UpdateByTwoPoints();
53                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
54                 helper2.BrushingPoints();
55                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
56                 await Navigation.PushAsync(new MainMenu());
57             }
58         }
59     }
60 }

```

```
58     }
59     /** 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
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
70 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.BuyOrganicFood"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Go Organic" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.FD.OrganicFood.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Purchase and consume organic food. It contains less
36 chemicals and is healthier than non organic food."
37 TextColor="Black"
38 HorizontalTextAlignment="Center"
39 FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="8 POINTS!"
44 TextColor="Black"
45 FontAttributes="Bold"
46 FontSize="20"
47 HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49 BackgroundColor="#50C878"
50 TextColor="White"
51 Margin="60,0,60,0"
52 Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```

```

1  /*! \class The BuyOrganicFood View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the BuyOrganicFood View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class BuyOrganicFood : ContentPage
20     {
21         public BuyOrganicFood()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.OrganicPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Travel.Carpool"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Carpool" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Travel.Carpool.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Carpool instead of going alone. This reduces emissions and
36 can be quiet fun too."
37 TextColor="Black"
38 HorizontalTextAlignment="Center"
39 FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="6 POINTS!"
44 TextColor="Black"
45 FontAttributes="Bold"
46 FontSize="20"
47 HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49 BackgroundColor="#50C878"
50 TextColor="White"
51 Margin="60,0,60,0"
52 Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```

```

1  /!* \class The Carpool View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the Carpool View Class. This class is the eco action that the user
   can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Travel
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class Carpool : ContentPage
20     {
21         public Carpool()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 TravelPointsUpdate helper2 = new TravelPointsUpdate();
53                 helper2.CarpoolPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```



```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Water.CisternDisplacement"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Cistern System" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Water.CisternDis.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Install a Cistern Displacement Device to use less water and
37 help reduce water waste."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="10 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>

```

```

1  /*! \class The CisternDisplacement View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the CisternDisplacement View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class CisternDisplacement : ContentPage
20     {
21         public CisternDisplacement()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
53                 helper2.CisternPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.ClothNapkins"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Cloth Napkins" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16
17     </NavigationPage.TitleView>
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Shopping.Napkin.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Purchase and use cloth napkins instead of paper ones. Paper
37 napkins contribute to waste and have to be remade. Cloth napkins can be washed and reused
38 and are biodegradable."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="2 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>

```

```

1  /!* \class The ClothNapkins View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ClothNapkins View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ClothNapkins : ContentPage
20     {
21         public ClothNapkins()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTwoPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.ClothNapkinsPoints();
54                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.ClothTowels"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Cloth Towels" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.Towel.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="2 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```



```

1  /*! \class The ClothTowels View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ClothTowels View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ClothTowels : ContentPage
20     {
21         public ClothTowels()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTwoPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.ClothTowelsPoints();
54                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Waste.CompostWaste"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Compost Bins" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16
17 </NavigationPage.TitleView>
18 <ContentPage.Content>
19     <ScrollView>
20         <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21             <Grid.RowDefinitions>
22                 <RowDefinition Height="2*"/>
23                 <RowDefinition Height="2*"/>
24                 <RowDefinition Height="*/>
25             </Grid.RowDefinitions>
26
27             <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Waste.Compost.jpg}"/>
30             <StackLayout Grid.Row="1"
31                 VerticalOptions="Center"
32                 HorizontalOptions="Center"
33                 Margin="15,0,15,0"
34                 Padding="0,5,0,5"
35                 BackgroundColor="#D3D3D3">
36                 <Label Text="Compost you food waste. Adding compost made from organic
37 materials to your garden improves moisture retention, provides slow-release nutrients and
38 reduce pesticide problem. Healthier plants grown from healthier soil are a huge and visible
39 bonus for all the keen gardeners out there."
40                 TextColor="Black"
41                 HorizontalTextAlignment="Center"
42                 FontSize="20"/>
43             </StackLayout>
44             <StackLayout Grid.Row="2"
45                 Spacing="10">
46                 <Label Text="6 POINTS!"
47                 TextColor="Black"
48                 FontAttributes="Bold"
49                 FontSize="20"
50                 HorizontalOptions="Center"/>
51                 <Button Text="Completed"
52                 BackgroundColor="#50C878"
53                 TextColor="White"
54                 Margin="60,0,60,0"
55                 Clicked="AddPointsClicked"/>
56             </StackLayout>
57         </Grid>
58     </ScrollView>
59 </ContentPage.Content>
60 </ContentPage>
```

```

1  /!* \class The CompostWaste View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the CompostWaste View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class CompostWaste : ContentPage
20     {
21         public CompostWaste()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 WastePointsUpdate helper2 = new WastePointsUpdate();
53                 helper2.CompostPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3           xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5 x:Class="Application_Green_Quake.Views.EcoActions.Community.CreateEnvironmentalGroup"
6           xmlns:local="clr-namespace:Application_Green_Quake.Models">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Create Group" TextColor="White" FontSize="20"
12            FontAttributes="Italic" VerticalOptions="CenterAndExpand"
13            HorizontalOptions="StartAndExpand"/>
14             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
15            VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
16         </StackLayout>
17     </NavigationPage.TitleView>
18
19     <ContentPage.Content>
20         <ScrollView>
21             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
22                 <Grid.RowDefinitions>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="2*"/>
25                     <RowDefinition Height="*/>
26                 </Grid.RowDefinitions>
27
28                 <Image Aspect="AspectFill"
29                    Source="{local:ImageResource
30                    Application_Green_Quake.Images.SubCategories.Community.CreateE.jpg}"/>
31                 <StackLayout Grid.Row="1"
32                    VerticalOptions="Center"
33                    HorizontalOptions="Center"
34                    Margin="15,0,15,0"
35                    Padding="0,5,0,5"
36                    BackgroundColor="#D3D3D3">
37                     <Label Text="No Environmental Group to join? Just create one and meet
38                    and gather like minded people!"
39                    TextColor="Black"
40                    HorizontalTextAlignment="Center"
41                    FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                    Spacing="10">
45                     <Label Text="10 POINTS!"
46                    TextColor="Black"
47                    FontAttributes="Bold"
48                    FontSize="20"
49                    HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51                    BackgroundColor="#50C878"
52                    TextColor="White"
53                    Margin="60,0,60,0"
54                    Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>

```

```

1  /*! \class The CreateEnvironmentalGroup View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the CreateEnvironmentalGroup View Class. This class is the eco
   action that the user can log.
7  *
8  */
9  using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Models;
11 using System;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14 using System.Threading.Tasks;
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class CreateEnvironmentalGroup : ContentPage
20     {
21         public CreateEnvironmentalGroup()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
53                 helper2.CreateGroupPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Travel.Cycle"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Cycle" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Travel.Cycle.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Cycle there if you can. Cycling is great for your overall
36 fitness and has no negative impacts on the environment."
37 TextColor="Black"
38 HorizontalTextAlignment="Center"
39 FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="10 POINTS!"
44 TextColor="Black"
45 FontAttributes="Bold"
46 FontSize="20"
47 HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49 BackgroundColor="#50C878"
50 TextColor="White"
51 Margin="60,0,60,0"
52 Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```

```

1  /!* \class The Cycle View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the Cycle View Class. This class is the eco action that the user can
log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Travel
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class Cycle : ContentPage
20     {
21         public Cycle()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 TravelPointsUpdate helper2 = new TravelPointsUpdate();
53                 helper2.CyclePoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59     * methods
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Habits.DishwasherFull"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Full Dishwasher" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16
17     </NavigationPage.TitleView>
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Habits.FullDishwasher.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Only use the dish washer when it is full. Believe it or not
37 dishwasher are more efficient than hand washing but only when they are loaded as they end up
38 using less water, less energy and more money."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="8 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>

```

```

1  /!* \class The DishwasherFull View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the DishwasherFull View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Habits
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class DishwasherFull : ContentPage
20     {
21         public DishwasherFull()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
53                 helper2.DishWasherFullPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Community.DoCommunity"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Community Work" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Community.DoCommunity.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Do something for the community and spread positivity. Small
37 things matter."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="10 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The DoCommunity View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the DoCommunity View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class DoCommunity : ContentPage
20     {
21         public DoCommunity()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
53                 helper2.CommunityPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```



```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Community.DonateItems"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Donate" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Community.Donate.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Instead of throwing items away or hoarding them it is
36 definitely a better option to give them away to the less fortunate. Not only will you be
37 doing a good deed but you will also feel great."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="10 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```
1 /!* \class The DonateItems View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the DonateItems View Class. This class is the eco action that the
   user can log.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class DonateItems : ContentPage
20     {
21         public DonateItems()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
53                 helper2.DonatePoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }
```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.DryerFull"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Full Dryer" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Energy.FullDryer.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Only use the dryer when it is full with clothes to save
36 energy. It is estimated that a dryer emits more than a ton of CO2 per year and contributes
37 to major energy waste. If possible skip using the dryer altogether."
38                     TextColor="Black"
39                     HorizontalTextAlignment="Center"
40                     FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="8 POINTS!"
45                     TextColor="Black"
46                     FontAttributes="Bold"
47                     FontSize="20"
48                     HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50                     BackgroundColor="#50C878"
51                     TextColor="White"
52                     Margin="60,0,60,0"
53                     Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The DryerFull View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the DryerFull View Class. This class is the eco action that the user
   can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class DryerFull : ContentPage
20     {
21         public DryerFull()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.DryerFullPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.EatAllYouMake"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Eat All" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.EatAllYouMake.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="4 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```



```

1  /*! \class The EatAllYouMake View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EatAllYouMake View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EatAllYouMake : ContentPage
20     {
21         public EatAllYouMake()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.EatAllPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3           xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5   x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EcoFreindlyAppliance"
6           xmlns:local="clr-namespace:Application_Green_Quake.Models">
7
8   <NavigationPage.TitleView>
9     <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10    VerticalOptions="EndAndExpand" Spacing="0">
11       <Label Text="Eco Appliance" TextColor="White" FontSize="20"
12      FontAttributes="Italic" VerticalOptions="CenterAndExpand"
13      HorizontalOptions="StartAndExpand"/>
14       <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
15      VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
16     </StackLayout>
17   </NavigationPage.TitleView>
18
19   <ContentPage.Content>
20     <ScrollView>
21       <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
22         <Grid.RowDefinitions>
23           <RowDefinition Height="2*"/>
24           <RowDefinition Height="2*"/>
25           <RowDefinition Height="*/>
26         </Grid.RowDefinitions>
27
28         <Image Aspect="AspectFill"
29          Source="{local:ImageResource
30      Application_Green_Quake.Images.SubCategories.Shopping.EcoAppliance.jpg}"/>
31         <StackLayout Grid.Row="1"
32          VerticalOptions="Center"
33          HorizontalOptions="Center"
34          Margin="15,0,15,0"
35          Padding="0,5,0,5"
36          BackgroundColor="#D3D3D3">
37           <Label Text="Purchase and use an Eco Friendly appliance. This uses less
38      energy and is better for the environment."
39          TextColor="Black"
40          HorizontalTextAlignment="Center"
41          FontSize="20"/>
42         </StackLayout>
43         <StackLayout Grid.Row="2"
44          Spacing="10">
45           <Label Text="4 POINTS!"
46          TextColor="Black"
47          FontAttributes="Bold"
48          FontSize="20"
49          HorizontalOptions="Center"/>
50           <Button Text="Completed"
51          BackgroundColor="#50C878"
52          TextColor="White"
53          Margin="60,0,60,0"
54          Clicked="AddPointsClicked"/>
55         </StackLayout>
56       </Grid>
57     </ScrollView>
58   </ContentPage.Content>
59 </ContentPage>

```

```

1  /*! \class The EcoFreidnlyApplicance View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EcoFreidnlyApplicance View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EcoFreidnlyApplicance : ContentPage
20     {
21         public EcoFreidnlyApplicance()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.AppliancePoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Travel.EcoFreindlyCar"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Eco Car" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Travel.EcoCar.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Purchase and use an Eco Friendly car. These types of cars
36 are more Environmentally Friendly that other cars. Just by simply owning one you are doing
37 more for the environment than people who do not own one."
38                         TextColor="Black"
39                         HorizontalTextAlignment="Center"
40                         FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="10 POINTS!"
45                         TextColor="Black"
46                         FontAttributes="Bold"
47                         FontSize="20"
48                         HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50                         BackgroundColor="#50C878"
51                         TextColor="White"
52                         Margin="60,0,60,0"
53                         Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The EcoFreindlyCar View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EcoFreindlyCar View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Travel
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EcoFreindlyCar : ContentPage
20     {
21         public EcoFreindlyCar()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 TravelPointsUpdate helper2 = new TravelPointsUpdate();
53                 helper2.EcoCarPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58 |         /** This function creates objects and calls their methods. First the security
    | methods are called and if they return false call the points updating
59 |             * methods
60 |             */
61 |         protected override void OnAppearing()
62 |         {
63 |             GetData data = new GetData();
64 |             data.SetLvl();
65 |
66 |             theLevel.Text = "LVL: " + GetData.lvl.ToString();
67 |         }
68 |     }
69 | }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EcoFriendlyProduct"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Eco Product" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Shopping.EcoProduct.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Purchase and use an Eco Friendly product that has no bad
37 chemicals and is manufactured in an eco friendlily way. This helps the environment and means
38 you are doing your part."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="4 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```

1  /*! \class The EcoFriendlyProduct View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EcoFriendlyProduct View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EcoFriendlyProduct : ContentPage
20     {
21         public EcoFriendlyProduct()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.ProductPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3           xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5   x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EcoFriendlyToothbrush"
6           xmlns:local="clr-namespace:Application_Green_Quake.Models">
7
8   <NavigationPage.TitleView>
9     <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10    VerticalOptions="EndAndExpand" Spacing="0">
11       <Label Text="Eco Toothbrush" TextColor="White" FontSize="20"
12      FontAttributes="Italic" VerticalOptions="CenterAndExpand"
13      HorizontalOptions="StartAndExpand"/>
14       <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
15      VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
16     </StackLayout>
17   </NavigationPage.TitleView>
18
19   <ContentPage.Content>
20     <ScrollView>
21       <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
22         <Grid.RowDefinitions>
23           <RowDefinition Height="2*"/>
24           <RowDefinition Height="2*"/>
25           <RowDefinition Height="*/>
26         </Grid.RowDefinitions>
27
28         <Image Aspect="AspectFill"
29          Source="{local:ImageResource
30      Application_Green_Quake.Images.SubCategories.Shopping.EcoBrush.jpg}"/>
31         <StackLayout Grid.Row="1"
32          VerticalOptions="Center"
33          HorizontalOptions="Center"
34          Margin="15,0,15,0"
35          Padding="0,5,0,5"
36          BackgroundColor="#D3D3D3">
37           <Label Text="Purchase and use an Eco Friendly toothbrush and toothpaste.
38      This is something we do at least twice a day so it is a good idea to do it in a Eco Friendly
39      Fashion."
40          TextColor="Black"
41          HorizontalTextAlignment="Center"
42          FontSize="20"/>
43         </StackLayout>
44         <StackLayout Grid.Row="2"
45          Spacing="10">
46           <Label Text="6 POINTS!"
47          TextColor="Black"
48          FontAttributes="Bold"
49          FontSize="20"
50          HorizontalOptions="Center"/>
51           <Button Text="Completed"
52          BackgroundColor="#50C878"
53          TextColor="White"
54          Margin="60,0,60,0"
55          Clicked="AddPointsClicked"/>
56         </StackLayout>
57       </Grid>
58     </ScrollView>
59   </ContentPage.Content>
60 </ContentPage>

```

```

1  /*! \class The EcoFriendlyToothbrush View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EcoFriendlyToothbrush View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EcoFriendlyToothbrush : ContentPage
20     {
21         public EcoFriendlyToothbrush()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.EcoToothbrushPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.EfficientThermostat"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Efficient Thermostat" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Energy.Thermostat.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Efficiently program your thermostat so it saves energy. Do
37 you ever have the heating on when you are not at home or the rooms are too warm?"
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="8 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The EfficientThermostat View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EfficientThermostat View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EfficientThermostat : ContentPage
20     {
21         public EfficientThermostat()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.EfficientThermostatPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```



```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3           xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5 x:Class="Application_Green_Quake.Views.EcoActions.Community.EnvironmentalGroups"
6           xmlns:local="clr-namespace:Application_Green_Quake.Models">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Join Group" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14            VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28                    Source="{local:ImageResource
29                    Application_Green_Quake.Images.SubCategories.Community.JoinE.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                    VerticalOptions="Center"
32                    HorizontalOptions="Center"
33                    Margin="15,0,15,0"
34                    Padding="0,5,0,5"
35                    BackgroundColor="#D3D3D3">
36                     <Label Text="Join an environmental group to discuss or to take action
37                    for the environment."
38                    TextColor="Black"
39                    HorizontalTextAlignment="Center"
40                    FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                    Spacing="10">
44                     <Label Text="8 POINTS!"
45                    TextColor="Black"
46                    FontAttributes="Bold"
47                    FontSize="20"
48                    HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50                    BackgroundColor="#50C878"
51                    TextColor="White"
52                    Margin="60,0,60,0"
53                    Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>

```

```

1  /*! \class The EnvironmentalGroups View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EnvironmentalGroups View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EnvironmentalGroups : ContentPage
20     {
21         public EnvironmentalGroups()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
53                 helper2.GroupPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EthicalClothes"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Ethical Clothes" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.EthicalClothes.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="10 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```

```

1  /*! \class The EthicalClothes View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the EthicalClothes View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EthicalClothes : ContentPage
20     {
21         public EthicalClothes()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.ClothesPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     xmlns:local="clr-namespace:Application_Green_Quake.Models"
5     x:Class="Application_Green_Quake.Views.EcoActions.AdvancedPageItems.FixInsteadOfThrowAway">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Fix It" TextColor="White" FontSize="20" FontAttributes="Italic"
11                 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13                 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28                         Application_Green_Quake.Images.SubCategories.Advanced.Fix.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Instead of throwing an item away try and fix it. This is
36                         much more environmentally friendly. If it cannot be fixed try and turn it into something
37                         else"
38                         TextColor="Black"
39                         HorizontalTextAlignment="Center"
40                         FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="10 POINTS!"
45                         TextColor="Black"
46                         FontAttributes="Bold"
47                         FontSize="20"
48                         HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50                         BackgroundColor="#50C878"
51                         TextColor="White"
52                         Margin="60,0,60,0"
53                         Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>

```



```

1  /*! \class The FixInsteadOfThrowAway View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the FixInsteadOfThrowAway View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.AdvancedPageItems
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class FixInsteadOfThrowAway : ContentPage
20     {
21         public FixInsteadOfThrowAway()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 AdvancedPointsUpdate helper2 = new AdvancedPointsUpdate();
53                 helper2.FixPoints();
54                 GetData data = new GetData();
55                 data.SetLvl();
56                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
57                 await Navigation.PushAsync(new MainMenu());

```

```
58     }
59   }
60   /** 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
61   * and display it in the navigation bar.
62   */
63   protected override void OnAppearing()
64   {
65     GetData data = new GetData();
66     data.SetLvl();
67
68     theLevel.Text = "LVL: " + GetData.lvl.ToString();
69   }
70 }
71 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.FoodDelivered"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Food Delivered" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.FD.FoodDelivered.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Instead of traveling to the shop have your food delivered
37 at once. This is more efficient and will result in less emissions as only a single delivery
38 vehicle will have to make a trip to all the houses rather than everyone going there and
39 back."
40                         TextColor="Black"
41                         HorizontalTextAlignment="Center"
42                         FontSize="20"/>
43                 </StackLayout>
44                 <StackLayout Grid.Row="2"
45                     Spacing="10">
46                     <Label Text="6 POINTS!"
47                         TextColor="Black"
48                         FontAttributes="Bold"
49                         FontSize="20"
50                         HorizontalOptions="Center"/>
51                     <Button Text="Completed"
52                         BackgroundColor="#50C878"
53                         TextColor="White"
54                         Margin="60,0,60,0"
55                         Clicked="AddPointsClicked"/>
56                 </StackLayout>
57             </Grid>
58         </ScrollView>
59     </ContentPage.Content>
60 </ContentPage>

```

```

1  /*! \class The FoodDelivered View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the FoodDelivered View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class FoodDelivered : ContentPage
20     {
21         public FoodDelivered()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.FoodDelivredPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.FoodInBulk"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Food In Bulk" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Shopping.FoodBulk.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Purchase Food in Bulk to save trips to the store in turn
37 saving energy and reducing emissions."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="6 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The FoodInBulk View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the FoodInBulk View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class FoodInBulk : ContentPage
20     {
21         public FoodInBulk()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.FoodInBulkPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```



```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.GoCamping"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Go Camping" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.Camping.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="6 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```

```

1  /!* \class The GoCamping View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the GoCamping View Class. This class is the eco action that the user
   can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class GoCamping : ContentPage
20     {
21         public GoCamping()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.CampingPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.HangDry"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Hang Dry" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Energy.HangDry.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Instead of using the dryer hang your clothes to dry when
36 you can. It may take longer, but it's much better on the environment, in more ways than
37 one."
38                         TextColor="Black"
39                         HorizontalTextAlignment="Center"
40                         FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="10 POINTS!"
45                         TextColor="Black"
46                         FontAttributes="Bold"
47                         FontSize="20"
48                         HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50                         BackgroundColor="#50C878"
51                         TextColor="White"
52                         Margin="60,0,60,0"
53                         Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The HangDry View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the HangDry View Class. This class is the eco action that the user
   can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class HangDry : ContentPage
20     {
21         public HangDry()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.HangDryPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.InsulateWater"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Water Tank" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.InsulateWaterTank.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="10 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>
```

```

1  /*! \class The InsulateWater View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the InsulateWater View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class InsulateWater : ContentPage
20     {
21         public InsulateWater()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.InsulateWaterPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```



```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.IsolateHome"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Insulate Home" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Energy.IsolateHome.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Insulate your home and keep the heat it. This will not only
37 make your home feel nice, hot and cozy but will also in turn save you money on heating."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="10 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The IsolateHome View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the IsolateHome View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class IsolateHome : ContentPage
20     {
21         public IsolateHome()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.IsolateHomePoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.LedLightBulb"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Led Lights" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Energy.LedLight.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Change to LED light bulbs now. They last longer than
36 conventional bulbs and are far more efficient."
37                         TextColor="Black"
38                         HorizontalTextAlignment="Center"
39                         FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="10 POINTS!"
44                         TextColor="Black"
45                         FontAttributes="Bold"
46                         FontSize="20"
47                         HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49                         BackgroundColor="#50C878"
50                         TextColor="White"
51                         Margin="60,0,60,0"
52                         Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```

```

1  /!* \class The LedLightBulb View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the LedLightBulb View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class LedLightBulb : ContentPage
20     {
21         public LedLightBulb()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.LedLightsPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.LocalProduct"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Buy Local" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Shopping.LocalProduct.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Buy things locally if possible and support your community.
36 Buying this locally is usually more environmentally friendly as the items you are purchasing
37 did not have to be imported. This also will help keep local businesses open."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="6 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```



```

1  /!* \class The LocalProduct View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the LocalProduct View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class LocalProduct : ContentPage
20     {
21         public LocalProduct()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.LocalProductPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58 | /** This function creates objects and calls their methods. First the security
    | methods are called and if they return false call the points updating
59 |     * methods
60 |     */
61 |     protected override void OnAppearing()
62 |     {
63 |         GetData data = new GetData();
64 |         data.SetLvl();
65 |
66 |         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67 |     }
68 | }
69 | }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.LooseLeafTea"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Loose Leaf" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Shopping.LooseTea.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Purchase and make loose leaf tea over bagged tea. Bagged
36 tea is pretty much pointless but ust more convenient even though it contributes to waste
37 with the bags. Loose leaf tea takes the bags out of the equation and even tastes better."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="4 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The LooseLeafTea View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the LooseLeafTea View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class LooseLeafTea : ContentPage
20     {
21         public LooseLeafTea()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.TeaPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.MachineFull"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Washing Machine" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Energy.FullWashingMachine.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Only use the washing machine when it is full. This will
37 save energy and also a ton of CO2 emission as a Washing Machine emits about 440kg of CO2 a
38 year on average."
39                         TextColor="Black"
40                         HorizontalTextAlignment="Center"
41                         FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="8 POINTS!"
46                         TextColor="Black"
47                         FontAttributes="Bold"
48                         FontSize="20"
49                         HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51                         BackgroundColor="#50C878"
52                         TextColor="White"
53                         Margin="60,0,60,0"
54                         Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>

```

```

1  /*! \class The MachineFull View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the MachineFull View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class MachineFull : ContentPage
20     {
21         public MachineFull()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.MachineFullPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.MicrowaveNotOven"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Microwave" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Energy.Microwave.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Microwave your food instead of heating it up using the oven
36 when you can and save energy. Plus it is faster."
37 TextColor="Black"
38 HorizontalTextAlignment="Center"
39 FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="4 POINTS!"
44 TextColor="Black"
45 FontAttributes="Bold"
46 FontSize="20"
47 HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49 BackgroundColor="#50C878"
50 TextColor="White"
51 Margin="60,0,60,0"
52 Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```

```

1  /*! \class The MicrowaveNotOven View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the MicrowaveNotOven View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class MicrowaveNotOven : ContentPage
20     {
21         public MicrowaveNotOven()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.MicrowavePoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.NoMeat"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="No Meat" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16     <ContentPage.Content>
17         <ScrollView>
18             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
19                 <Grid.RowDefinitions>
20                     <RowDefinition Height="2*"/>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="*/>
23                 </Grid.RowDefinitions>
24
25                 <Image Aspect="AspectFill"
26 Source="{local:ImageResource
27 Application_Green_Quake.Images.SubCategories.FD.NoMeat.jpg}"/>
28                 <StackLayout Grid.Row="1"
29                     VerticalOptions="Center"
30                     HorizontalOptions="Center"
31                     Margin="15,0,15,0"
32                     Padding="0,5,0,5"
33                     BackgroundColor="#D3D3D3">
34                     <Label Text="Don't eat meat today. This will seriously reduce greenhouse
35 gas emissions."
36                         TextColor="Black"
37                         HorizontalTextAlignment="Center"
38                         FontSize="20"/>
39                 </StackLayout>
40                 <StackLayout Grid.Row="2"
41                     Spacing="10">
42                     <Label Text="10 POINTS!"
43                         TextColor="Black"
44                         FontAttributes="Bold"
45                         FontSize="20"
46                         HorizontalOptions="Center"/>
47                     <Button Text="Completed"
48                         BackgroundColor="#50C878"
49                         TextColor="White"
50                         Margin="60,0,60,0"
51                         Clicked="AddPointsClicked"/>
52                 </StackLayout>
53             </Grid>
54         </ScrollView>
55     </ContentPage.Content>
56 </ContentPage>
```

```

1  /!* \class The NoMeat View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the NoMeat View Class. This class is the eco action that the user
   can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class NoMeat : ContentPage
20     {
21         public NoMeat()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.NoMeatPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Home.NonHarmfulProducts"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6     <NavigationPage.TitleView>
7         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
8 VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Non Harmful" TextColor="White" FontSize="20"
10 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
11 HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Home.NonHarmful.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Purchase and use non harmful products. These reduce
36 pollution levels and are healthier."
37 TextColor="Black"
38 HorizontalTextAlignment="Center"
39 FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="4 POINTS!"
44 TextColor="Black"
45 FontAttributes="Bold"
46 FontSize="20"
47 HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49 BackgroundColor="#50C878"
50 TextColor="White"
51 Margin="60,0,60,0"
52 Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```

```

1  /!* \class The NonHarmfulProducts View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the NonHarmfulProducts View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class NonHarmfulProducts : ContentPage
20     {
21         public NonHarmfulProducts()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 HomePointsUpdate helper2 = new HomePointsUpdate();
53                 helper2.NonHarmfulPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```



```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Work.OffElectronics"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Electronics Off" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Work.Off.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Turn off electronics that are not in use. There is no need
37 to pointlessly waste electricity."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="6 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1 using Application_Green_Quake.Models;
2 using Application_Green_Quake.ViewModels;
3 using System;
4 using System.Threading.Tasks;
5 using Xamarin.Forms;
6 using Xamarin.Forms.Xaml;
7
8 namespace Application_Green_Quake.Views.EcoActions.Work
9 {
10     [XamlCompilation(XamlCompilationOptions.Compile)]
11     public partial class OffElectronics : ContentPage
12     {
13         public OffElectronics()
14         {
15             InitializeComponent();
16             OnAppearing();
17         }
18         /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
19         * methods
20         */
21         private async void AddPointsClicked(object sender, EventArgs e)
22         {
23             SecurityMethods checks = new SecurityMethods();
24             Task<bool> myTask = checks.DayLimitLock();
25             await myTask;
26
27             Task<bool> myTaskTwo = checks.TimeLimitLock();
28             await myTaskTwo;
29
30             if (myTask.Result)
31             {
32                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
33                 await Navigation.PushAsync(new MainMenu());
34             }
35             else if (myTaskTwo.Result)
36             {
37                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
38                 await Navigation.PushAsync(new MainMenu());
39             }
40             else
41             {
42                 PointsUpdate helper = new PointsUpdate();
43                 helper.UpdateBySixPoints();
44                 WorkPointsUpdate helper2 = new WorkPointsUpdate();
45                 helper2.ElectonicsOffPoints();
46                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
47                 await Navigation.PushAsync(new MainMenu());
48             }
49         }
50         /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
51         * methods
52         */
53         protected override void OnAppearing()
54         {
55             GetData data = new GetData();
56             data.SetLvl();
57

```

```
58 |         theLevel.Text = "LVL: " + GetData.lvl.ToString();  
59 |     }  
60 | }  
61 | }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.OffSocketSwitch"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Socket Off" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.OffSocket.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="4 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```

```

1  /!* \class The OffSocketSwitch View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the OffSocketSwitch View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OffSocketSwitch : ContentPage
20     {
21         public OffSocketSwitch()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.SocketPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.OrganicFood"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Go Organic" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.FD.OrganicFood.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Purchase and consume organic food. It contains less
36 chemicals and is healthier than non organic food."
37 TextColor="Black"
38 HorizontalTextAlignment="Center"
39 FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="6 POINTS!"
44 TextColor="Black"
45 FontAttributes="Bold"
46 FontSize="20"
47 HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49 BackgroundColor="#50C878"
50 TextColor="White"
51 Margin="60,0,60,0"
52 Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```



```

1  /*! \class The OrganicFood View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the OrganicFood View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OrganicFood : ContentPage
20     {
21         public OrganicFood()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.OrganicPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58 | /** This function creates objects and calls their methods. First the security  
    | methods are called and if they return false call the points updating  
59 |     * methods  
60 |     */  
61 |     protected override void OnAppearing()  
62 |     {  
63 |         GetData data = new GetData();  
64 |         data.SetLvl();  
65 |  
66 |         theLevel.Text = "LVL: " + GetData.lvl.ToString();  
67 |     }  
68 | }  
69 | }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Home.OutsideOnce"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Go Outside" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Home.Outside.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Go outside once a day and get that Vitamin D!. Going
36 outside at least once a day is good for your mental and physical health. In addition no
37 electricity is being used by you when you are outside."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="2 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The OutsideOnce View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the OutsideOnce View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OutsideOnce : ContentPage
20     {
21         public OutsideOnce()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTwoPoints();
52                 HomePointsUpdate helper2 = new HomePointsUpdate();
53                 helper2.OutsidePoints();
54                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.OwnCoffee"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Brew Coffee" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16
17 </NavigationPage.TitleView>
18 <ContentPage.Content>
19     <ScrollView>
20         <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21             <Grid.RowDefinitions>
22                 <RowDefinition Height="2*"/>
23                 <RowDefinition Height="2*"/>
24                 <RowDefinition Height="*/>
25             </Grid.RowDefinitions>
26
27             <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.FD.OwnCoffee.jpg}"/>
30             <StackLayout Grid.Row="1"
31                 VerticalOptions="Center"
32                 HorizontalOptions="Center"
33                 Margin="15,0,15,0"
34                 Padding="0,5,0,5"
35                 BackgroundColor="#D3D3D3">
36                 <Label Text="Purchasing coffee and making your own is better for the
37 environment that going to the coffee shop."
38                 TextColor="Black"
39                 HorizontalTextAlignment="Center"
40                 FontSize="20"/>
41             </StackLayout>
42             <StackLayout Grid.Row="2"
43                 Spacing="10">
44                 <Label Text="2 POINTS!"
45                 TextColor="Black"
46                 FontAttributes="Bold"
47                 FontSize="20"
48                 HorizontalOptions="Center"/>
49                 <Button Text="Completed"
50                 BackgroundColor="#50C878"
51                 TextColor="White"
52                 Margin="60,0,60,0"
53                 Clicked="AddPointsClicked"/>
54             </StackLayout>
55         </Grid>
56     </ScrollView>
57 </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The OwnCoffee View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the OwnCoffee View Class. This class is the eco action that the user
   can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OwnCoffee : ContentPage
20     {
21         public OwnCoffee()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTwoPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.OwnCoffeePoints();
54                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.Picnic"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Have A Picnic" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Outdoors.Picnic.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Go out for a picnic and use reusable cutlery. This can be
37 very positive for you mood."
38                         TextColor="Black"
39                         HorizontalTextAlignment="Center"
40                         FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="6 POINTS!"
45                         TextColor="Black"
46                         FontAttributes="Bold"
47                         FontSize="20"
48                         HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50                         BackgroundColor="#50C878"
51                         TextColor="White"
52                         Margin="60,0,60,0"
53                         Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The Picnic View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the Picnic View Class. This class is the eco action that the user
   can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class Picnic : ContentPage
20     {
21         public Picnic()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.PicnicPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.PlantABush"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Plant A Bush" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Outdoors.Bush.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Plant a bush where you can. As time goes on the planet is
37 losing more and more of it's greenery. Planting a bush can decrease this loss and it helps
38 the environment by producing oxygen."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="8 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```

1  /!* \class The PlantABush View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the PlantABush View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PlantABush : ContentPage
20     {
21         public PlantABush()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.PlantBushPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.PlantAFlower"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Plant A Flower" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Outdoors.Flower.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Plant flowers where you can. This does not only improve the
37 environment but also looks visually appealing and smells nice."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="8 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The PlantAFlower View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the PlantAFlower View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PlantAFlower : ContentPage
20     {
21         public PlantAFlower()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.PlantFlowerPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```



```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.PlantATree"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Plant A Tree" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Outdoors.Tree.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Plant a tree where you can. The main environmental effects
37 of deforestation and forest degradation include reduced biodiversity, the release of
38 greenhouse gas emissions, forest fires, disrupted water cycles and increased soil erosion."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="10 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```

1  /*! \class The PlantATree View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the PlantATree View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PlantATree : ContentPage
20     {
21         public PlantATree()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.PlantTreePoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Home.PlantIntoHome"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Plant Inside" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Home.PlantHome.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Bring a plant into your home. A plant cleans indoor air by
37 absorbing toxins, increasing humidity and producing oxygen."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="4 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The PlantIntoHome View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the PlantIntoHome View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PlantIntoHome : ContentPage
20     {
21         public PlantIntoHome()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 HomePointsUpdate helper2 = new HomePointsUpdate();
53                 helper2.PlantsInsidePoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Travel.PublicTransport"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Public Transport" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Travel.PublicTransport.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Use public transport when ou cannot cycle or walk as it is
37 even more efficient than carpooling or driving."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="8 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```



```

1  /!* \class The PublicTransport View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the PublicTransport View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Travel
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PublicTransport : ContentPage
20     {
21         public PublicTransport()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 TravelPointsUpdate helper2 = new TravelPointsUpdate();
53                 helper2.TransportPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3           xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5   x:Class="Application_Green_Quake.Views.EcoActions.Shopping.PurchaseReusableWater"
6           xmlns:local="clr-namespace:Application_Green_Quake.Models">
7
8   <NavigationPage.TitleView>
9     <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10    VerticalOptions="EndAndExpand" Spacing="0">
11       <Label Text="Reusable Bottle" TextColor="White" FontSize="20"
12      FontAttributes="Italic" VerticalOptions="CenterAndExpand"
13      HorizontalOptions="StartAndExpand"/>
14       <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
15      VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
16     </StackLayout>
17   </NavigationPage.TitleView>
18
19   <ContentPage.Content>
20     <ScrollView>
21       <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
22         <Grid.RowDefinitions>
23           <RowDefinition Height="2*"/>
24           <RowDefinition Height="2*"/>
25           <RowDefinition Height="*/>
26         </Grid.RowDefinitions>
27
28         <Image Aspect="AspectFill"
29          Source="{local:ImageResource
30      Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"/>
31         <StackLayout Grid.Row="1"
32          VerticalOptions="Center"
33          HorizontalOptions="Center"
34          Margin="15,0,15,0"
35          Padding="0,5,0,5"
36          BackgroundColor="#D3D3D3">
37           <Label Text="Instead of using and buying plastic bottles use a reusable
38      water bottle and refill it each time. This removes your plastic bottle waste. At this rate
39      99 million tons of plastic waste will end up in the environment by 2030"
40          TextColor="Black"
41          HorizontalTextAlignment="Center"
42          FontSize="20"/>
43         </StackLayout>
44         <StackLayout Grid.Row="2"
45          Spacing="10">
46           <Label Text="6 POINTS!"
47          TextColor="Black"
48          FontAttributes="Bold"
49          FontSize="20"
50          HorizontalOptions="Center"/>
51           <Button Text="Completed"
52          BackgroundColor="#50C878"
53          TextColor="White"
54          Margin="60,0,60,0"
55          Clicked="AddPointsClicked"/>
56         </StackLayout>
57       </Grid>
58     </ScrollView>
59   </ContentPage.Content>
60 </ContentPage>

```

```

1  /*! \class The PurchaseReusableWater View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the PurchaseReusableWater View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PurchaseReusableWater : ContentPage
20     {
21         public PurchaseReusableWater()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.ReWaterPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59     * methods
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Water.RainBarrel"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Rain Barrel" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Water.RainBarrel.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <Label Text="Set up a rain barrel to collect rain water. This rainwater
can then be reused for other things such as watering plants."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="10 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```

```

1  /!* \class The RainBarrel View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the RainBarrel View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class RainBarrel : ContentPage
20     {
21         public RainBarrel()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
53                 helper2.BarrelPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.ReBatteries"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Reusable Batteries" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Energy.ReBatteries.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Purchase, use and reuse rechargeable batteries instead of
37 regular batteries. This prevents us from having to make as many as we do and reduces the
38 waste produced by them."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="6 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```

1  /*! \class The ReBatteries View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ReBatteries View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ReBatteries : ContentPage
20     {
21         public ReBatteries()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
53                 helper2.ReBattereisPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.ReCoffeeMug"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Reusable Cup" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12
13    </NavigationPage.TitleView>
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.ReCoffeeMug.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="4 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```

```

1  /*! \class The ReCoffeeMug View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ReCoffeeMug View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ReCoffeeMug : ContentPage
20     {
21         public ReCoffeeMug()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.ReCoffeeMugPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.RefrigeratorDown"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Refrigerator" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16
17 </NavigationPage.TitleView>
18 <ContentPage.Content>
19     <ScrollView>
20         <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21             <Grid.RowDefinitions>
22                 <RowDefinition Height="2*"/>
23                 <RowDefinition Height="2*"/>
24                 <RowDefinition Height="*/>
25             </Grid.RowDefinitions>
26
27             <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Energy.TurnDownFridge.jpg}"/>
30             <StackLayout Grid.Row="1"
31                 VerticalOptions="Center"
32                 HorizontalOptions="Center"
33                 Margin="15,0,15,0"
34                 Padding="0,5,0,5"
35                 BackgroundColor="#D3D3D3">
36                 <Label Text="Turn down the refrigerator. Your fridge might be on a
37 higher than needed setting. Turn this down and save energy. The halocarbons in refrigeration
38 appliances contribute to the greenhouse effect which effects the ozone layer and contributes
39 to global warming."
40                 TextColor="Black"
41                 HorizontalTextAlignment="Center"
42                 FontSize="20"/>
43             </StackLayout>
44             <StackLayout Grid.Row="2"
45                 Spacing="10">
46                 <Label Text="8 POINTS!"
47                 TextColor="Black"
48                 FontAttributes="Bold"
49                 FontSize="20"
50                 HorizontalOptions="Center"/>
51                 <Button Text="Completed"
52                 BackgroundColor="#50C878"
53                 TextColor="White"
54                 Margin="60,0,60,0"
55                 Clicked="AddPointsClicked"/>
56             </StackLayout>
57         </Grid>
58     </ScrollView>
59 </ContentPage.Content>
60 </ContentPage>
```

```
1 /!* \class The RefrigeratorDown View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the RefrigeratorDown View Class. This class is the eco action that
   the user can log.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class RefrigeratorDown : ContentPage
20     {
21         public RefrigeratorDown()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.FridgePoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }
```



```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Shopping.ReusableBag"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Reusable Bag" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.ReBag.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <Label Text="Always use a reusable bag. Using paper or plastic bags for
single use add to waste."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="8 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```

```
1 /!* \class The ReusableBag View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the ReusableBag View Class. This class is the eco action that the
   user can log.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ReusableBag : ContentPage
20     {
21         public ReusableBag()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26
27         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
42                 await Navigation.PushAsync(new MainMenu());
43             }
44             else if (myTaskTwo.Result)
45             {
46                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
47                 await Navigation.PushAsync(new MainMenu());
48             }
49             else
50             {
51                 PointsUpdate helper = new PointsUpdate();
52                 helper.UpdateBySixPoints();
53                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
54                 helper2.ReWaterPoints();
55                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
56                 await Navigation.PushAsync(new MainMenu());
57             }
58         }
59     }
60 }
```

```
58     }
59     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
60         * methods
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
70 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Water.ReusableWater"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Reusable Bottle" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*" />
19                    <RowDefinition Height="2*" />
20                    <RowDefinition Height="*" />
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="6 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>
```

```

1  /*! \class The ReusableWater View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ReusableWater View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ReusableWater : ContentPage
20     {
21         public ReusableWater()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
53                 helper2.ReWaterPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.SaveLeftOvers"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Save Leftovers" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12
13    </NavigationPage.TitleView>
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.SaveLeftovers.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="6 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```



```

1  /*! \class The SaveLeftOvers View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SaveLeftOvers View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SaveLeftOvers : ContentPage
20     {
21         public SaveLeftOvers()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateBySixPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.SaveLeftOversPoints();
54                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.Scoop"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Scoop da Poop" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Outdoors.Scoop.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="When you dog makes a mess clean it up. It is your duty as a
37 dog owner"
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="4 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The Scoop View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the Scoop View Class. This class is the eco action that the user can
log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class Scoop : ContentPage
20     {
21         public Scoop()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.ScoopPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.SealDrafts"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Seal Draft" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16     <ContentPage.Content>
17         <ScrollView>
18             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
19                 <Grid.RowDefinitions>
20                     <RowDefinition Height="2*"/>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="*/>
23                 </Grid.RowDefinitions>
24
25                 <Image Aspect="AspectFill"
26 Source="{local:ImageResource
27 Application_Green_Quake.Images.SubCategories.Energy.SealDraft.jpg}"/>
28                 <StackLayout Grid.Row="1"
29                     VerticalOptions="Center"
30                     HorizontalOptions="Center"
31                     Margin="15,0,15,0"
32                     Padding="0,5,0,5"
33                     BackgroundColor="#D3D3D3">
34                     <Label Text="Seal the drafts in your home and keep it warm. This will
35 save you a ton of energy as you will not have to use the heating as frequently."
36 TextColor="Black"
37 HorizontalTextAlignment="Center"
38 FontSize="20"/>
39                 </StackLayout>
40                 <StackLayout Grid.Row="2"
41                     Spacing="10">
42                     <Label Text="10 POINTS!"
43 TextColor="Black"
44 FontAttributes="Bold"
45 FontSize="20"
46 HorizontalOptions="Center"/>
47                     <Button Text="Completed"
48 BackgroundColor="#50C878"
49 TextColor="White"
50 Margin="60,0,60,0"
51 Clicked="AddPointsClicked"/>
52                 </StackLayout>
53             </Grid>
54         </ScrollView>
55     </ContentPage.Content>
56 </ContentPage>
```

```
1 /!* \class The SealDrafts View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the SealDrafts View Class. This class is the eco action that the
   user can log.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SealDrafts : ContentPage
20     {
21         public SealDrafts()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.SealDraftsPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }
```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.SealDucts"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Seal Duct" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16     <ContentPage.Content>
17         <ScrollView>
18             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
19                 <Grid.RowDefinitions>
20                     <RowDefinition Height="2*"/>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="*/>
23                 </Grid.RowDefinitions>
24
25                 <Image Aspect="AspectFill"
26 Source="{local:ImageResource
27 Application_Green_Quake.Images.SubCategories.Energy.SealDuct.jpg}"/>
28                 <StackLayout Grid.Row="1"
29                     VerticalOptions="Center"
30                     HorizontalOptions="Center"
31                     Margin="15,0,15,0"
32                     Padding="0,5,0,5"
33                     BackgroundColor="#D3D3D3">
34                     <Label Text="Seal the ducts in your home and keep it warm. This will
35 save you a ton of energy as you will not have to use the heating as frequently."
36 TextColor="Black"
37 HorizontalTextAlignment="Center"
38 FontSize="20"/>
39                 </StackLayout>
40                 <StackLayout Grid.Row="2"
41                     Spacing="10">
42                     <Label Text="8 POINTS!"
43 TextColor="Black"
44 FontAttributes="Bold"
45 FontSize="20"
46 HorizontalOptions="Center"/>
47                     <Button Text="Completed"
48 BackgroundColor="#50C878"
49 TextColor="White"
50 Margin="60,0,60,0"
51 Clicked="AddPointsClicked"/>
52                 </StackLayout>
53             </Grid>
54         </ScrollView>
55     </ContentPage.Content>
56 </ContentPage>
```

```

1  /!* \class The SealDucts View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SealDucts View Class. This class is the eco action that the user
   can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SealDucts : ContentPage
20     {
21         public SealDucts()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.SealDuctsPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.SetUpFruitGarden"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Fruit Garden" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Outdoors.FruitGarden.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Set up a fruit garden. This is great for the environment
37 and makes you self sustainable and you will no longer need to purchase some fruits."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="10 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The SetUpFruitGarden View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SetUpFruitGarden View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SetUpFruitGarden : ContentPage
20     {
21         public SetUpFruitGarden()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.FruitGardenPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.SetUpHerbGarden"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Herb Garden" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Outdoors.HerbGarden.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Set up a Herb garden. This is great for the environment and
37 makes you self sustainable and you will no longer need to purchase some herbs."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="10 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The SetUpHerbGarden View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SetUpHerbGarden View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SetUpHerbGarden : ContentPage
20     {
21         public SetUpHerbGarden()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.HerbGardenPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```



```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Waste.SetUpRecyclingBin"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Recycling Bins" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Waste.SetUpBins.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Set up recycling bins in you home. This reduces the amount
37 of waste sent to landfills and incinerators. Conserves natural resources such as timber,
38 water and minerals and saves energy."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44 Spacing="10">
45                     <Label Text="10 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```

1  /*! \class The SetUpRecyclingBin View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SetUpRecyclingBin View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SetUpRecyclingBin : ContentPage
20     {
21         public SetUpRecyclingBin()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 WastePointsUpdate helper2 = new WastePointsUpdate();
53                 helper2.SetUpRecyclingBinPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3           xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5 x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.SetUpVegetableGarden"
6           xmlns:local="clr-namespace:Application_Green_Quake.Models">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Vegetable Garden" TextColor="White" FontSize="20"
12            FontAttributes="Italic" VerticalOptions="CenterAndExpand"
13            HorizontalOptions="StartAndExpand"/>
14             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
15            VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
16         </StackLayout>
17     </NavigationPage.TitleView>
18
19     <ContentPage.Content>
20         <ScrollView>
21             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
22                 <Grid.RowDefinitions>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="2*"/>
25                     <RowDefinition Height="*/>
26                 </Grid.RowDefinitions>
27
28                 <Image Aspect="AspectFill"
29                 Source="{local:ImageResource
30                Application_Green_Quake.Images.SubCategories.Outdoors.VegGarden.jpg}"/>
31                 <StackLayout Grid.Row="1"
32                 VerticalOptions="Center"
33                 HorizontalOptions="Center"
34                 Margin="15,0,15,0"
35                 Padding="0,5,0,5"
36                 BackgroundColor="#D3D3D3">
37                     <Label Text="Set up a Vegetable garden. This is great for the
38                    environment and makes you self sustainable and you will no longer need to purchase some
39                    vegetables."
40                    TextColor="Black"
41                    HorizontalTextAlignment="Center"
42                    FontSize="20"/>
43                 </StackLayout>
44                 <StackLayout Grid.Row="2"
45                 Spacing="10">
46                     <Label Text="10 POINTS!"
47                    TextColor="Black"
48                    FontAttributes="Bold"
49                    FontSize="20"
50                    HorizontalOptions="Center"/>
51                     <Button Text="Completed"
52                    BackgroundColor="#50C878"
53                    TextColor="White"
54                    Margin="60,0,60,0"
55                    Clicked="AddPointsClicked"/>
56                 </StackLayout>
57             </Grid>
58         </ScrollView>
59     </ContentPage.Content>
60 </ContentPage>

```

```

1  /!* \class The SetUpVegetableGarden View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SetUpVegetableGarden View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SetUpVegetableGarden : ContentPage
20     {
21         public SetUpVegetableGarden()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.VegetableGardenPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Community.ShareThisApp"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Share App" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Community.Share.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="This one is simple! Share this app and let's grow
36 together!"
37                         TextColor="Black"
38                         HorizontalTextAlignment="Center"
39                         FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="10 POINTS!"
44                         TextColor="Black"
45                         FontAttributes="Bold"
46                         FontSize="20"
47                         HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49                         BackgroundColor="#50C878"
50                         TextColor="White"
51                         Margin="60,0,60,0"
52                         Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```



```

1  /!* \class The ShareThisApp View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ShareThisApp View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ShareThisApp : ContentPage
20     {
21         public ShareThisApp()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
53                 helper2.SharePoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Water.ShowerBucket"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Shower Bucket" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Water.ShowerBucket.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Place a shower bucket in your shower to collect the water
37 you use. This water can be used for other things such as watering plants"
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="8 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The ShowerBucket View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ShowerBucket View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ShowerBucket : ContentPage
20     {
21         public ShowerBucket()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
53                 helper2.ShowerBucketPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Habits.ShowerInstead"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Shower No Bath" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16
17     </NavigationPage.TitleView>
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Habits.ShowerNoBath.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Shower instead of taking a bath. This is more efficient. If
37 you don't take too long of course. The average bath uses 36 gallons to fill a tub, while the
38 average shower (without the water-saving device) uses five gallons of water per minute."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="6 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```

1  /!* \class The ShowerInstead View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ShowerInstead View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13
14 using Xamarin.Forms;
15 using Xamarin.Forms.Xaml;
16
17 namespace Application_Green_Quake.Views.EcoActions.Habits
18 {
19     [XamlCompilation(XamlCompilationOptions.Compile)]
20     public partial class ShowerInstead : ContentPage
21     {
22         public ShowerInstead()
23         {
24             InitializeComponent();
25             OnAppearing();
26         }
27         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
42                 await Navigation.PushAsync(new MainMenu());
43             }
44             else if (myTaskTwo.Result)
45             {
46                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
47                 await Navigation.PushAsync(new MainMenu());
48             }
49             else
50             {
51                 PointsUpdate helper = new PointsUpdate();
52                 helper.UpdateBySixPoints();
53                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
54                 helper2.ShowerInsteadPoints();
55                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
56                 await Navigation.PushAsync(new MainMenu());
57             }
58         }
59     }
60 }

```

```
58     }
59     /** 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
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
70 }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Energy.SolarPanel"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Solar Panel" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Energy.SolarPanel.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Install a Solar Panel. Using solar energy can have a
37 positive, indirect effect on the environment when solar energy replaces or reduces the use
38 of other energy sources that have larger effects on the environment."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="10 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```

1  /*! \class The SolarPanel View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SolarPanel View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SolarPanel : ContentPage
20     {
21         public SolarPanel()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
53                 helper2.SolarPanelPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Community.SpreadAwareness"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Spread Awareness" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Community.Spread.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Spread Awareness about the environment by posting online
37 and the alike. 77% of people would like to live more sustainably and you could be helping
38 them by showing how."
39 TextColor="Black"
40 HorizontalTextAlignment="Center"
41 FontSize="20"/>
42                 </StackLayout>
43                 <StackLayout Grid.Row="2"
44                     Spacing="10">
45                     <Label Text="10 POINTS!"
46 TextColor="Black"
47 FontAttributes="Bold"
48 FontSize="20"
49 HorizontalOptions="Center"/>
50                     <Button Text="Completed"
51 BackgroundColor="#50C878"
52 TextColor="White"
53 Margin="60,0,60,0"
54 Clicked="AddPointsClicked"/>
55                 </StackLayout>
56             </Grid>
57         </ScrollView>
58     </ContentPage.Content>
59 </ContentPage>
```

```

1  /*! \class The SpreadAwareness View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SpreadAwareness View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SpreadAwareness : ContentPage
20     {
21         public SpreadAwareness()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
53                 helper2.awarenessPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.SteelStraw"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Steel Straws" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16
17 </NavigationPage.TitleView>
18 <ContentPage.Content>
19     <ScrollView>
20         <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21             <Grid.RowDefinitions>
22                 <RowDefinition Height="2*"/>
23                 <RowDefinition Height="2*"/>
24                 <RowDefinition Height="*/>
25             </Grid.RowDefinitions>
26
27             <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.FD.SteeStraw.jpg}"/>
30             <StackLayout Grid.Row="1"
31                 VerticalOptions="Center"
32                 HorizontalOptions="Center"
33                 Margin="15,0,15,0"
34                 Padding="0,5,0,5"
35                 BackgroundColor="#D3D3D3">
36                 <Label Text="Purchase and use steel straws over regular straws. This
37 means you can reuse them and reduces waste. In just the U.S. alone, one estimate suggests
38 500 million straws are used every single day."
39                 FontSize="20"
40                 TextColor="Black"/>
41             </StackLayout>
42             <StackLayout Grid.Row="2"
43                 Spacing="10">
44                 <Label Text="4 POINTS!"
45                 TextColor="Black"
46                 FontAttributes="Bold"
47                 FontSize="20"
48                 HorizontalOptions="Center"/>
49                 <Button Text="Completed"
50                 BackgroundColor="#50C878"
51                 TextColor="White"
52                 Margin="60,0,60,0"
53                 Clicked="AddPointsClicked"/>
54             </StackLayout>
55         </Grid>
56     </ScrollView>
57 </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The SteelStraw View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SteelStraw View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SteelStraw : ContentPage
20     {
21         public SteelStraw()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.SteelStrawPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```



```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Habits.TimedShower"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Timed Shower" TextColor="White" FontSize="20"
FontAttributes="Italic" VerticalOptions="CenterAndExpand"
HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.TimedShower.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="4 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```

```

1  /*! \class The TimedShower View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the TimedShower View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Habits
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class TimedShower : ContentPage
20     {
21         public TimedShower()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
53                 helper2.TimedShowerInsteadPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Home.ToiletFlushes"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Save A Flush" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Home.Flush.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Instead of flushing the toilet every time try and flush
37 only when it is necessary to save water."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="4 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The ToiletFlushes View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the ToiletFlushes View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ToiletFlushes : ContentPage
20     {
21         public ToiletFlushes()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 HomePointsUpdate helper2 = new HomePointsUpdate();
53                 helper2.ToiletPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Habits.TurnOffLights"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Turn Off Lights" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16
17 </NavigationPage.TitleView>
18 <ContentPage.Content>
19     <ScrollView>
20         <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21             <Grid.RowDefinitions>
22                 <RowDefinition Height="2*"/>
23                 <RowDefinition Height="2*"/>
24                 <RowDefinition Height="*/>
25             </Grid.RowDefinitions>
26
27             <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Habits.OffLights.jpg}"/>
30             <StackLayout Grid.Row="1"
31                 VerticalOptions="Center"
32                 HorizontalOptions="Center"
33                 Margin="15,0,15,0"
34                 Padding="0,5,0,5"
35                 BackgroundColor="#D3D3D3">
36                 <Label Text="Turn off the lights when leaving the room. If you turn off
37 the lights whenever you leave a room, you can reduce greenhouse gas emissions by 0.15 pounds
38 per hour."
39                 TextColor="Black"
40                 HorizontalTextAlignment="Center"
41                 FontSize="20"/>
42             </StackLayout>
43             <StackLayout Grid.Row="2"
44                 Spacing="10">
45                 <Label Text="2 POINTS!"
46                 TextColor="Black"
47                 FontAttributes="Bold"
48                 FontSize="20"
49                 HorizontalOptions="Center"/>
50                 <Button Text="Completed"
51                 BackgroundColor="#50C878"
52                 TextColor="White"
53                 Margin="60,0,60,0"
54                 Clicked="AddPointsClicked"/>
55             </StackLayout>
56         </Grid>
57     </ScrollView>
58 </ContentPage.Content>
59 </ContentPage>

```



```
1 /!* \class The TurnOffLights View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the TurnOffLights View Class. This class is the eco action that the
   user can log.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13
14 using Xamarin.Forms;
15 using Xamarin.Forms.Xaml;
16
17 namespace Application_Green_Quake.Views.EcoActions.Habits
18 {
19     [XamlCompilation(XamlCompilationOptions.Compile)]
20     public partial class TurnOffLights : ContentPage
21     {
22         public TurnOffLights()
23         {
24             InitializeComponent();
25             OnAppearing();
26         }
27         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
42                 await Navigation.PushAsync(new MainMenu());
43             }
44             else if (myTaskTwo.Result)
45             {
46                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
47                 await Navigation.PushAsync(new MainMenu());
48             }
49             else
50             {
51                 PointsUpdate helper = new PointsUpdate();
52                 helper.UpdateByTwoPoints();
53                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
54                 helper2.OffLightsPoints();
55                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
56                 await Navigation.PushAsync(new MainMenu());
57             }
58         }
59     }
60 }
```

```
58     }
59     /** 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
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
70 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Outdoors.UpBirdfeeder"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Bird Feeder" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Outdoors.BirdFeeder.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Set up a bird feeder and help the birds. Support
37 biodiversity by supporting other animals"
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="10 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The UpBirdfeeder View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the UpBirdfeeder View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class UpBirdfeeder : ContentPage
20     {
21         public UpBirdfeeder()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
53                 helper2.BirdFeederPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58 | /** This function creates objects and calls their methods. First the security
    | methods are called and if they return false call the points updating
59 |     * methods
60 |     */
61 |     protected override void OnAppearing()
62 |     {
63 |         GetData data = new GetData();
64 |         data.SetLvl();
65 |
66 |         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67 |     }
68 | }
69 | }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Waste.UseBiogradableBinBags"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Bio Bin Bags" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Waste.BioBags.jpg}"/>
30                 <StackLayout Grid.Row="1"
31 VerticalOptions="Center"
32 HorizontalOptions="Center"
33 Margin="15,0,15,0"
34 Padding="0,5,0,5"
35 BackgroundColor="#D3D3D3">
36                     <Label Text="Use biodegradable bin bags as these can rot and don't
37 pollute the environment."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43 Spacing="10">
44                     <Label Text="4 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /*! \class The UseBiogradableBinBags View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the UseBiogradableBinBags View Class. This class is the eco action
   that the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class UseBiogradableBinBags : ContentPage
20     {
21         public UseBiogradableBinBags()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByFourPoints();
52                 WastePointsUpdate helper2 = new WastePointsUpdate();
53                 helper2.BioBinBagPoints();
54                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```



```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Habits.UseMatches"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="Matches" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12    </NavigationPage.TitleView>
13
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.MatchOverLighter.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="2 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>

```

```

1  /!* \class The UseMatches View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the UseMatches View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Habits
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class UseMatches : ContentPage
20     {
21         public UseMatches()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTwoPoints();
52                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
53                 helper2.MatchesPoints();
54                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Waste.UseRecyclingBin"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Recycle" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Waste.Recycled.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Recycle when you can! This reduces the amount of waste sent
36 to landfills and incinerators. Conserves natural resources such as timber, water and
37 minerals and saves energy."
38                         TextColor="Black"
39                         HorizontalTextAlignment="Center"
40                         FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="10 POINTS!"
45                         TextColor="Black"
46                         FontAttributes="Bold"
47                         FontSize="20"
48                         HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50                         BackgroundColor="#50C878"
51                         TextColor="White"
52                         Margin="60,0,60,0"
53                         Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```

1  /!* \class The UseRecyclingBin View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the UseRecyclingBin View Class. This class is the eco action that
   the user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class UseRecyclingBin : ContentPage
20     {
21         public UseRecyclingBin()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 WastePointsUpdate helper2 = new WastePointsUpdate();
53                 helper2.RecyclingBinPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }

```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Travel.Walk"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Walk" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Travel.Walk.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Walk there if you can as this is the most environmentally
36 friendly from of transportation."
37 TextColor="Black"
38 HorizontalTextAlignment="Center"
39 FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="10 POINTS!"
44 TextColor="Black"
45 FontAttributes="Bold"
46 FontSize="20"
47 HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49 BackgroundColor="#50C878"
50 TextColor="White"
51 Margin="60,0,60,0"
52 Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```

```

1  /!* \class The Walk View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the Walk View Class. This class is the eco action that the user can
   log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Travel
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class Walk : ContentPage
20     {
21         public Walk()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 TravelPointsUpdate helper2 = new TravelPointsUpdate();
53                 helper2.WalkPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```



```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.WaterOverFizzy"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
VerticalOptions="EndAndExpand" Spacing="0">
9             <Label Text="H2O" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
10            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
11        </StackLayout>
12
13    </NavigationPage.TitleView>
14    <ContentPage.Content>
15        <ScrollView>
16            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
17                <Grid.RowDefinitions>
18                    <RowDefinition Height="2*"/>
19                    <RowDefinition Height="2*"/>
20                    <RowDefinition Height="*/>
21                </Grid.RowDefinitions>
22
23                <Image Aspect="AspectFill"
24                    Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.WaterOverFiz.jpg}"/>
25                <StackLayout Grid.Row="1"
26                    VerticalOptions="Center"
27                    HorizontalOptions="Center"
28                    Margin="15,0,15,0"
29                    Padding="0,5,0,5"
30                    BackgroundColor="#D3D3D3">
31                    <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."
32                        TextColor="Black"
33                        HorizontalTextAlignment="Center"
34                        FontSize="20"/>
35                </StackLayout>
36                <StackLayout Grid.Row="2"
37                    Spacing="10">
38                    <Label Text="8 POINTS!"
39                        TextColor="Black"
40                        FontAttributes="Bold"
41                        FontSize="20"
42                        HorizontalOptions="Center"/>
43                    <Button Text="Completed"
44                        BackgroundColor="#50C878"
45                        TextColor="White"
46                        Margin="60,0,60,0"
47                        Clicked="AddPointsClicked"/>
48                </StackLayout>
49            </Grid>
50        </ScrollView>
51    </ContentPage.Content>
52 </ContentPage>
```

```

1  /*! \class The WaterOverFizzy View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the WaterOverFizzy View Class. This class is the eco action that the
   user can log.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class WaterOverFizzy : ContentPage
20     {
21         public WaterOverFizzy()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByEightPoints();
52                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
53                 helper2.WaterOverFizzyPoints();
54                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }

```

```
58     /** 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
59     * and display it in the navigation bar.
60     */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Work.WorkingRemotely"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="Remote Work" TextColor="White" FontSize="20"
11 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12 HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28 Source="{local:ImageResource
29 Application_Green_Quake.Images.SubCategories.Work.Remote.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="If possible work remotely. Not only is this more relaxing
37 and less time consuming but it also saves travel costs and reduces emissions."
38 TextColor="Black"
39 HorizontalTextAlignment="Center"
40 FontSize="20"/>
41                 </StackLayout>
42                 <StackLayout Grid.Row="2"
43                     Spacing="10">
44                     <Label Text="10 POINTS!"
45 TextColor="Black"
46 FontAttributes="Bold"
47 FontSize="20"
48 HorizontalOptions="Center"/>
49                     <Button Text="Completed"
50 BackgroundColor="#50C878"
51 TextColor="White"
52 Margin="60,0,60,0"
53 Clicked="AddPointsClicked"/>
54                 </StackLayout>
55             </Grid>
56         </ScrollView>
57     </ContentPage.Content>
58 </ContentPage>
```

```
1 using Application_Green_Quake.Models;
2 using Application_Green_Quake.ViewModels;
3 using System;
4 using System.Threading.Tasks;
5 using Xamarin.Forms;
6 using Xamarin.Forms.Xaml;
7
8 namespace Application_Green_Quake.Views.EcoActions.Work
9 {
10     [XamlCompilation(XamlCompilationOptions.Compile)]
11     public partial class WorkingRemotely : ContentPage
12     {
13         public WorkingRemotely()
14         {
15             InitializeComponent();
16             OnAppearing();
17         }
18         /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
19         * methods
20         */
21         private async void AddPointsClicked(object sender, EventArgs e)
22         {
23             SecurityMethods checks = new SecurityMethods();
24             Task<bool> myTask = checks.DayLimitLock();
25             await myTask;
26
27             Task<bool> myTaskTwo = checks.TimeLimitLock();
28             await myTaskTwo;
29
30             if (myTask.Result)
31             {
32                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
day.", "OK");
33                 await Navigation.PushAsync(new MainMenu());
34             }
35             else if (myTaskTwo.Result)
36             {
37                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
next Action.", "OK");
38                 await Navigation.PushAsync(new MainMenu());
39             }
40             else
41             {
42                 PointsUpdate helper = new PointsUpdate();
43                 helper.UpdateByTenPoints();
44                 WorkPointsUpdate helper2 = new WorkPointsUpdate();
45                 helper2.RemoteWorkPoints();
46                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
47                 await Navigation.PushAsync(new MainMenu());
48             }
49         }
50         /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
51         * methods
52         */
53         protected override void OnAppearing()
54         {
55             GetData data = new GetData();
56             data.SetLvl();
57         }
58     }
59 }
```

```
58 |         theLevel.Text = "LVL: " + GetData.lvl.ToString();  
59 |     }  
60 | }  
61 | }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.Water.WSShowerHead"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9 VerticalOptions="EndAndExpand" Spacing="0">
10             <Label Text="WSS Head" TextColor="White" FontSize="20" FontAttributes="Italic"
11 VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13 VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14         </StackLayout>
15     </NavigationPage.TitleView>
16
17     <ContentPage.Content>
18         <ScrollView>
19             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                 <Grid.RowDefinitions>
21                     <RowDefinition Height="2*"/>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="*/>
24                 </Grid.RowDefinitions>
25
26                 <Image Aspect="AspectFill"
27 Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Water.ShowerHead.jpg}"/>
29                 <StackLayout Grid.Row="1"
30                     VerticalOptions="Center"
31                     HorizontalOptions="Center"
32                     Margin="15,0,15,0"
33                     Padding="0,5,0,5"
34                     BackgroundColor="#D3D3D3">
35                     <Label Text="Install a Water saving Shower Head to use less water when
36 you shower."
37                         TextColor="Black"
38                         HorizontalTextAlignment="Center"
39                         FontSize="20"/>
40                 </StackLayout>
41                 <StackLayout Grid.Row="2"
42                     Spacing="10">
43                     <Label Text="10 POINTS!"
44                         TextColor="Black"
45                         FontAttributes="Bold"
46                         FontSize="20"
47                         HorizontalOptions="Center"/>
48                     <Button Text="Completed"
49                         BackgroundColor="#50C878"
50                         TextColor="White"
51                         Margin="60,0,60,0"
52                         Clicked="AddPointsClicked"/>
53                 </StackLayout>
54             </Grid>
55         </ScrollView>
56     </ContentPage.Content>
57 </ContentPage>
```



```
1 /!* \class The WSShowerHead View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the WSShowerHead View Class. This class is the eco action that the
   user can log.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using System;
12 using System.Threading.Tasks;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class WSShowerHead : ContentPage
20     {
21         public WSShowerHead()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
   methods are called and if they return false call the points updating
27         * methods
28         */
29         private async void AddPointsClicked(object sender, EventArgs e)
30         {
31             SecurityMethods checks = new SecurityMethods();
32             Task<bool> myTask = checks.DayLimitLock();
33             await myTask;
34
35             Task<bool> myTaskTwo = checks.TimeLimitLock();
36             await myTaskTwo;
37
38             if (myTask.Result)
39             {
40                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
   day.", "OK");
41                 await Navigation.PushAsync(new MainMenu());
42             }
43             else if (myTaskTwo.Result)
44             {
45                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
   next Action.", "OK");
46                 await Navigation.PushAsync(new MainMenu());
47             }
48             else
49             {
50                 PointsUpdate helper = new PointsUpdate();
51                 helper.UpdateByTenPoints();
52                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
53                 helper2.WSSowerHeadPoints();
54                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
55                 await Navigation.PushAsync(new MainMenu());
56             }
57         }
58     }
59 }
```

```
58     /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59         * methods
60         */
61     protected override void OnAppearing()
62     {
63         GetData data = new GetData();
64         data.SetLvl();
65
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();
67     }
68 }
69 }
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <pages:PopupPage x:Class="Application_Green_Quake.Views.LeaderboardPage.LeaderBoardPopUp"
3     xmlns="http://xamarin.com/schemas/2014/forms"
4     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5     xmlns:animations="clr-
namespace:Rg.Plugins.Popup.Animations;assembly=Rg.Plugins.Popup"
6     xmlns:pages="clr-
namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
7     BackgroundColor="#002a1e">
8
9     <StackLayout HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" >
10         <Label x:Name="theName" TextColor="White" FontSize="20"
HorizontalTextAlignment="Center"/>
11         <Image x:Name="profileImage" HeightRequest="300"/>
12         <Label x:Name="theRank" TextColor="White" FontSize="20"
HorizontalTextAlignment="Center"/>
13         <Label x:Name="theBio" TextColor="White" FontSize="20"
HorizontalTextAlignment="Center"/>
14         <Label x:Name="thePoints" TextColor="White" FontSize="20"
HorizontalTextAlignment="Center"/>
15     </StackLayout>
16
17 </pages:PopupPage>
```

```
1 /!* \class The LeaderBoardPopUp View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the LeaderBoardPopUp View Class. This class is the popup that
   appears when a entry in the LeaderBoard is tapped.
7 *
8 */
9 using Xamarin.Forms;
10 using Xamarin.Forms.Xaml;
11
12 namespace Application_Green_Quake.Views.LeaderboardPage
13 {
14     [XamlCompilation(XamlCompilationOptions.Compile)]
15     public partial class LeaderBoardPopUp
16     {
17         /** The LeaderBoardPopUp Constructor
18         @param username is the username to display
19         @param points are the points to display
20         @param rank is the rank to display
21         @param image is the image to display
22         @param bio is the bio to display
23         */
24         public LeaderBoardPopUp(string username, int points, string rank, ImageSource image,
   string bio)
25         {
26             InitializeComponent();
27             profileImage.Source = image;
28             theName.Text = "Username: " + username;
29             thePoints.Text = "Points: " + points.ToString();
30             theBio.Text = "Bio: " + bio;
31             theRank.Text = rank;
32         }
33     }
34 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     xmlns:models="clr-namespace:Application_Green_Quake.Models;assembly=Application
Green Quake"
5     xmlns:xForms="clr-
namespace:Syncfusion.SfPicker.XForms;assembly=Syncfusion.SfPicker.XForms"
6     x:Class="Application_Green_Quake.Views.LeadersboardPage.TopTabLeaderBoard">
7     <ContentPage.Content>
8         <StackLayout BackgroundColor="#002a1e">
9             <xForms:SfPicker x:Name="picker"
10                 ItemHeight="45"
11                 HeaderText="Filter by Country"
12                 PickerMode="Dialog"
13                 PickerHeight="350"
14                 PickerWidth="350"
15                 ShowFooter="True"
16                 OkButtonClicked="PickerOnOkButtonClicked"
17                 HeaderBackgroundColor="#50C878"
18                 SelectedItemTextColor="#50C878"
19                 OKButtonTextColor="#50C878"
20                 CancelButtonTextColor="#50C878"/>
21             <StackLayout Orientation="Horizontal" Margin="15,5,15,5">
22                 <Label Text="Leader Board" FontSize="22" HorizontalTextAlignment="Center"
FontFamily="Proxima Nova" TextColor="White"/>
23                 <Image Source="{models:ImageResource
Application_Green_Quake.Images.filter.png}" HorizontalOptions="EndAndExpand">
24                     <Image.GestureRecognizers>
25                         <TapGestureRecognizer Tapped="ImageClicked"/>
26                     </Image.GestureRecognizers>
27                 </Image>
28             </StackLayout>
29             <ListView x:Name="LeaderBoard"
30                 ItemTapped="OnItemTapped"
31                 RowHeight="70">
32                 <ListView.ItemTemplate>
33                     <DataTemplate>
34                         <ViewCell>
35                             <StackLayout BackgroundColor="White">
36                                 <Grid Margin="20,0,20,0">
37                                     <Grid.ColumnDefinitions>
38                                         <ColumnDefinition Width="*" />
39                                         <ColumnDefinition Width="2*" />
40                                         <ColumnDefinition Width="2*" />
41                                     </Grid.ColumnDefinitions>
42                                     <Grid.RowDefinitions>
43                                         <RowDefinition Height="*" />
44                                     </Grid.RowDefinitions>
45
46                                     <Frame Padding="0" Margin="0,5,0,5" CornerRadius="80">
47                                         <Image Source="{Binding image}"
Aspect="AspectFill"/>
48                                     </Frame>
49
50                                     <StackLayout Grid.Column="1" Margin="10,5,0,0"
Spacing="0">
51                                         <Label Text="{Binding username}" TextColor="Black"
FontAttributes="Bold" FontSize="15" FontFamily="Roboto"/>
52                                         <Label Text="{Binding points, StringFormat='Score:
{0}'}" FontSize="12" TextColor="Silver" FontFamily="Roboto"/>
53                                     </StackLayout>
54                                     <StackLayout Grid.Column="2">

```

```
55         <Label Text="{Binding rank}" Margin="0,5,0,0"
HorizontalOptions="End" FontFamily="Roboto" TextColor="Silver"/>
56     </StackLayout>
57     </Grid>
58 </StackLayout>
59 </ViewCell>
60 </DataTemplate>
61 </ListView.ItemTemplate>
62 </ListView>
63 <StackLayout Orientation="Horizontal" Margin="15,5,15,5">
64     <Label Text="Back To Page 1" FontSize="22" FontFamily="Proxima Nova"
TextColor="White" HorizontalOptions="Start" HeightRequest="30">
65         <Label.GestureRecognizers>
66             <TapGestureRecognizer Tapped="FirstPageClicked"/>
67         </Label.GestureRecognizers>
68     </Label>
69     <Image Source="{models:ImageResource
Application_Green_Quake.Images.right.png}" HorizontalOptions="EndAndExpand"
HeightRequest="30">
70         <Image.GestureRecognizers>
71             <TapGestureRecognizer Tapped="NextPageClicked"/>
72         </Image.GestureRecognizers>
73     </Image>
74 </StackLayout>
75 </StackLayout>
76 </ContentPage.Content>
77 </ContentPage>
```

```
1 /!* \class The TopTabLeaderBoard View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the TopTabLeaderBoard View Class. This is the class for the
   leaderboard screen.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Firebase.Database;
11 using Firebase.Database.Query;
12 using Firebase.Storage;
13 using Rg.Plugins.Popup.Services;
14 using System;
15 using System.Collections.ObjectModel;
16 using System.Linq;
17 using Acr.UserDialogs;
18 using Application_Green_Quake.ViewModels;
19 using Xamarin.Forms;
20 using Xamarin.Forms.Xaml;
21 using SelectionChangedEventArgs = Syncfusion.SfPicker.XForms.SelectionChangedEventArgs;
22
23 namespace Application_Green_Quake.Views.LeaderboardPage
24 {
25     [XamlCompilation(XamlCompilationOptions.Compile)]
26     public partial class TopTabLeaderBoard : ContentPage
27     {
28         private string selectedNation = "All";
29         private int min = 0;
30         private int count = 10;
31         IAuth auth;
32         public TopTabLeaderBoard()
33         {
34             InitializeComponent();
35             ObservableCollection<object> countryNameCollection = new
36             ObservableCollection<object>();
37
38             // Add al the options into the filter
39             countryNameCollection.Add("All");
40             countryNameCollection.Add("Me");
41             countryNameCollection.Add("Albania");
42             countryNameCollection.Add("Andorra");
43             countryNameCollection.Add("Armenia");
44             countryNameCollection.Add("Austria");
45             countryNameCollection.Add("Azerbaijan");
46             countryNameCollection.Add("Belarus");
47             countryNameCollection.Add("Belgium");
48             countryNameCollection.Add("Bosnia and Herzegovina");
49             countryNameCollection.Add("Bulgaria");
50             countryNameCollection.Add("Croatia");
51             countryNameCollection.Add("Cyprus");
52             countryNameCollection.Add("Czech Republic");
53             countryNameCollection.Add("Denmark");
54             countryNameCollection.Add("Estonia");
55             countryNameCollection.Add("Finland");
56             countryNameCollection.Add("France");
57             countryNameCollection.Add("Georgia");
58             countryNameCollection.Add("Germany");
59             countryNameCollection.Add("Greece");
```

```

59     countryNameCollection.Add("Hungary");
60     countryNameCollection.Add("Iceland");
61     countryNameCollection.Add("Ireland");
62     countryNameCollection.Add("Italy");
63     countryNameCollection.Add("Kazakhstan");
64     countryNameCollection.Add("Kosovo");
65     countryNameCollection.Add("Latvia");
66     countryNameCollection.Add("Liechtenstein");
67     countryNameCollection.Add("Lithuania");
68     countryNameCollection.Add("Luxembourg");
69     countryNameCollection.Add("Malta");
70     countryNameCollection.Add("Moldova");
71     countryNameCollection.Add("Monaco");
72     countryNameCollection.Add("Montenegro");
73     countryNameCollection.Add("Netherlands");
74     countryNameCollection.Add("North Macedonia");
75     countryNameCollection.Add("Norway");
76     countryNameCollection.Add("Poland");
77     countryNameCollection.Add("Portugal");
78     countryNameCollection.Add("Romania");
79     countryNameCollection.Add("Russia");
80     countryNameCollection.Add("San Marino");
81     countryNameCollection.Add("Serbia");
82     countryNameCollection.Add("Slovakia");
83     countryNameCollection.Add("Slovenia");
84     countryNameCollection.Add("Spain");
85     countryNameCollection.Add("Sweden");
86     countryNameCollection.Add("Switzerland");
87     countryNameCollection.Add("Turkey");
88     countryNameCollection.Add("Ukraine");
89     countryNameCollection.Add("United Kingdom");
90     countryNameCollection.Add("Vatican City");
91
92     picker.ItemsSource = countryNameCollection;
93     auth = DependencyService.Get<IAuth>();
94     OnAppearing();
95 }
96 /** This function is called before the page is displayed.
97 */
98 protected override async void OnAppearing()
99 {
100     //If All gets selected from the filter then display all the profiles
101     if (selectedNation == "All")
102     {
103         UserDialogs.Instance.ShowLoading();
104         FirebaseClient firebaseClient = new FirebaseClient("https://application-
green-quake-default-rtdb.firebaseio.com/");
105
106         //Save the data into a list and order it by points
107         var list = (await firebaseClient
108             .Child("Points")
109             .OnceAsync<Points>()).Select(item => new Points
110             {
111                 username = item.Object.username,
112                 points = item.Object.points,
113             }).ToList().OrderByDescending(s => s.points);
114
115         //Save that data to a second list
116         var list2 = list.Select(item => new LeaderBoard
117             {
118                 username = item.username,

```



```

119         points = item.points,
120     }).ToList();
121
122     int index = 0;
123     string rankIndex = "";
124     // Create a new list and assign an image and the rank based on the index
125     foreach (var i in list2)
126     {
127         index++;
128         rankIndex = "Rank: " + index.ToString();
129         i.rank = rankIndex;
130         try
131         {
132             var uid = (await firebaseClient
133                 .Child("usernames")
134                 .Child(i.username)
135                 .OnceSingleAsync<Usernames>()).Uid;
136
137             i.image = await new FirebaseStorage("application-green-
quake.appspot.com")
138                 .Child(uid)
139                 .Child("Profile.jpg")
140                 .GetDownloadUrlAsync();
141         }
142         catch (Exception e)
143         {
144             i.image =
ImageSource.FromResource("Application_Green_Quake.Images.user.png");
145             Console.Write(e);
146         }
147         try
148         {
149             var uid = (await firebaseClient
150                 .Child("usernames")
151                 .Child(i.username)
152                 .OnceSingleAsync<Usernames>()).Uid;
153
154             i.bio = (await firebaseClient
155                 .Child("users")
156                 .Child(uid)
157                 .OnceSingleAsync<Users>()).bio;
158
159             i.nation = (await firebaseClient
160                 .Child("users")
161                 .Child(uid)
162                 .OnceSingleAsync<Users>()).nation;
163         }
164         catch (Exception e)
165         {
166             Console.Write(e);
167         }
168     }
169
170     try
171     { //Get entries between specified indexes and dsave them into a list and
then display them in the leaderboard.
172         var list3 = list2.Select(item => new LeaderBoard
173             {
174                 username = item.username,
175                 points = item.points,
176                 nation = item.nation,
177                 bio = item.nation,

```

```

178         rank = item.rank,
179         image = item.image
180     }).ToList().GetRange(min, count);
181     LeaderBoard.ItemsSource = list3;
182
183     }
184     catch (Exception e)
185     {
186         try
187         { //Get entries between specified indexes and dsave them into a list
and then display them in the leaderboard.
188             var list3 = list2.Select(item => new LeaderBoard
189             {
190                 username = item.username,
191                 points = item.points,
192                 nation = item.nation,
193                 bio = item.bio,
194                 rank = item.rank,
195                 image = item.image
196             }).ToList().GetRange(min, list2.Count - min);
197             LeaderBoard.ItemsSource = list3;
198         }
199         catch (Exception exception)
200         {
201             min = 0;
202             await DisplayAlert("Last Page", "You have reached the last leader
board page.", "Ok");
203         }
204     }
205     UserDialogs.Instance.HideLoading();
206 }
207 //If Me gets selected from the filter then only display the users profile
208 else if (selectedNation == "Me")
209 {
210     UserDialogs.Instance.ShowLoading();
211     FirebaseClient firebaseClient = new FirebaseClient("https://application-
green-quake-default-rtdb.firebaseio.com/");
212
213     var list = (await firebaseClient
214         .Child("Points")
215         .OnceAsync<Points>()).Select(item => new Points
216     {
217         username = item.Object.username,
218         points = item.Object.points,
219     }).ToList().OrderByDescending(s => s.points);
220
221     var list2 = list.Select(item => new LeaderBoard
222     {
223         username = item.username,
224         points = item.points,
225     }).ToList();
226
227     int index = 0;
228     string rankIndex = "";
229     foreach (var i in list2)
230     {
231         index++;
232         rankIndex = "Rank: " + index.ToString();
233         i.rank = rankIndex;
234         try
235         {
236             var uid = (await firebaseClient

```

```

237         .Child("usernames")
238         .Child(i.username)
239         .OnceSingleAsync<Usernames>()).Uid;
240
241         i.uid = uid;
242
243         i.image = await new FirebaseStorage("application-green-
quake.appspot.com")
244         .Child(uid)
245         .Child("Profile.jpg")
246         .GetDownloadUrlAsync();
247     }
248     catch (Exception e)
249     {
250         i.image =
ImageSource.FromResource("Application_Green_Quake.Images.user.png");
251         Console.WriteLine(e);
252     }
253     try
254     {
255         var uid = (await firebaseClient
256         .Child("usernames")
257         .Child(i.username)
258         .OnceSingleAsync<Usernames>()).Uid;
259
260         i.uid = uid;
261
262         i.bio = (await firebaseClient
263         .Child("users")
264         .Child(uid)
265         .OnceSingleAsync<Users>()).bio;
266
267         i.nation = (await firebaseClient
268         .Child("users")
269         .Child(uid)
270         .OnceSingleAsync<Users>()).nation;
271     }
272     catch (Exception e)
273     {
274         Console.WriteLine(e);
275     }
276 }
277
278 var list3 = list2.Select(item => new LeaderBoard
279 {
280     username = item.username,
281     points = item.points,
282     nation = item.nation,
283     bio = item.nation,
284     rank = item.rank,
285     image = item.image,
286     uid = item.uid
287 }).Where(x => x.uid == auth.GetUid()).ToList();
288
289 LeaderBoard.ItemsSource = list3;
290 UserDialogs.Instance.ShowLoading();
291 }
292 else
293 {
294     UserDialogs.Instance.ShowLoading();
295     FirebaseClient firebaseClient = new FirebaseClient("https://application-
green-quake-default-rtdb.firebaseio.com/");

```

```
296
297     var list = (await firebaseClient
298         .Child("Points")
299         .OnceAsync<Points>()).Select(item => new Points
300     {
301         username = item.Object.username,
302         points = item.Object.points,
303     }).ToList().OrderByDescending(s => s.points);
304
305     var list2 = list.Select(item => new LeaderBoard
306     {
307         username = item.username,
308         points = item.points,
309     }).ToList();
310
311     int index = 0;
312     string rankIndex = "";
313     foreach (var i in list2)
314     {
315         index++;
316         rankIndex = "Rank: " + index.ToString();
317         i.rank = rankIndex;
318         try
319         {
320             var uid = (await firebaseClient
321                 .Child("usernames")
322                 .Child(i.username)
323                 .OnceSingleAsync<Usernames>()).Uid;
324
325             i.image = await new FirebaseStorage("application-green-
326 quake.appspot.com")
327                 .Child(uid)
328                 .Child("Profile.jpg")
329                 .GetDownloadUrlAsync();
330         }
331         catch (Exception e)
332         {
333             i.image =
334             ImageSource.FromResource("Application_Green_Quake.Images.user.png");
335             Console.Write(e);
336         }
337         try
338         {
339             var uid = (await firebaseClient
340                 .Child("usernames")
341                 .Child(i.username)
342                 .OnceSingleAsync<Usernames>()).Uid;
343
344             i.bio = (await firebaseClient
345                 .Child("users")
346                 .Child(uid)
347                 .OnceSingleAsync<Users>()).bio;
348
349             i.nation = (await firebaseClient
350                 .Child("users")
351                 .Child(uid)
352                 .OnceSingleAsync<Users>()).nation;
353         }
354         catch (Exception e)
355         {
356             Console.Write(e);
357         }
358     }
```

```

356     }
357
358     var list3 = list2.Select(item => new LeaderBoard
359     {
360         username = item.username,
361         points = item.points,
362         nation = item.nation,
363         bio = item.bio,
364         rank = item.rank,
365         image = item.image
366     }).Where(n => n.nation == selectedNation).ToList();
367
368     index = 0;
369     rankIndex = "";
370     foreach (var p in list3)
371     {
372         index++;
373         rankIndex = "Rank: " + index.ToString();
374         p.rank = rankIndex;
375     }
376
377     LeaderBoard.ItemsSource = list3;
378 }
379 //Stop the loading spinner and set the data.
380 UserDialogs.Instance.HideLoading();
381 GetData data = new GetData();
382 data.SetLvl();
383 min = 0;
384 count = 10;
385 }
386
387 /** This function displays the correct popup when a profile on the leader board is
tapped.
388 */
389 private void OnItemTapped (Object sender, ItemTappedEventArgs e)
390 {
391
392     var dataItem = e.Item as LeaderBoard;
393     PopupNavigation.Instance.PushAsync(new LeaderBoardPopUp(dataItem.username,
dataItem.points, dataItem.rank, dataItem.image, dataItem.bio));
394 }
395 /** This function displays the picker when the icon is pressed.
396 */
397 private async void ImageClicked(object sender, EventArgs e)
398 {
399     picker.IsOpen = true;
400 }
401
402 /** This function sets the selectedNation variable when a value is selected on the
picker and ok is pressed..
403 */
404 private async void PickerOnOkButtonClicked(object sender, SelectionChangedEventArgs
e)
405 {
406     if (picker.SelectedItem.ToString() == "All")
407     {
408         selectedNation = "All";
409     }
410     else if (picker.SelectedItem.ToString() == "Me")
411     {
412         selectedNation = "Me";
413     }

```

```
414     else if (picker.SelectedItem.ToString() == "Albania")
415     {
416         selectedNation = "Albania";
417     }
418     else if (picker.SelectedItem.ToString() == "Andorra")
419     {
420         selectedNation = "Andorra";
421     }
422     else if (picker.SelectedItem.ToString() == "Armenia")
423     {
424         selectedNation = "Armenia";
425     }
426     else if (picker.SelectedItem.ToString() == "Austria")
427     {
428         selectedNation = "Austria";
429     }
430     else if (picker.SelectedItem.ToString() == "Azerbaijan")
431     {
432         selectedNation = "Azerbaijan";
433     }
434     else if (picker.SelectedItem.ToString() == "Belarus")
435     {
436         selectedNation = "Belarus";
437     }
438     else if (picker.SelectedItem.ToString() == "Belgium")
439     {
440         selectedNation = "Belgium";
441     }
442     else if (picker.SelectedItem.ToString() == "Bosnia and Herzegovina")
443     {
444         selectedNation = "Bosnia and Herzegovina";
445     }
446     else if (picker.SelectedItem.ToString() == "Bulgaria")
447     {
448         selectedNation = "Bulgaria";
449     }
450     else if (picker.SelectedItem.ToString() == "Cyprus")
451     {
452         selectedNation = "Cyprus";
453     }
454     else if (picker.SelectedItem.ToString() == "Czech Republic")
455     {
456         selectedNation = "Czech Republic";
457     }
458     else if (picker.SelectedItem.ToString() == "Denmark")
459     {
460         selectedNation = "Denmark";
461     }
462     else if (picker.SelectedItem.ToString() == "Estonia")
463     {
464         selectedNation = "Estonia";
465     }
466     else if (picker.SelectedItem.ToString() == "Finland")
467     {
468         selectedNation = "Finland";
469     }
470     else if (picker.SelectedItem.ToString() == "France")
471     {
472         selectedNation = "France";
473     }
474     else if (picker.SelectedItem.ToString() == "Georgia")
```

```
475     {
476         selectedNation = "Georgia";
477     }
478     else if (picker.SelectedItem.ToString() == "Germany")
479     {
480         selectedNation = "Germany";
481     }
482     else if (picker.SelectedItem.ToString() == "Greece")
483     {
484         selectedNation = "Greece";
485     }
486     else if (picker.SelectedItem.ToString() == "Hungary")
487     {
488         selectedNation = "Hungary";
489     }
490     else if (picker.SelectedItem.ToString() == "Iceland")
491     {
492         selectedNation = "Iceland";
493     }
494     else if (picker.SelectedItem.ToString() == "Ireland")
495     {
496         selectedNation = "Ireland";
497     }
498     else if (picker.SelectedItem.ToString() == "Italy")
499     {
500         selectedNation = "Italy";
501     }
502     else if (picker.SelectedItem.ToString() == "Kazakhstan")
503     {
504         selectedNation = "Kazakhstan";
505     }
506     else if (picker.SelectedItem.ToString() == "Kosovo")
507     {
508         selectedNation = "Kosovo";
509     }
510     else if (picker.SelectedItem.ToString() == "Latvia")
511     {
512         selectedNation = "Latvia";
513     }
514     else if (picker.SelectedItem.ToString() == "Liechtenstein")
515     {
516         selectedNation = "Liechtenstein";
517     }
518     else if (picker.SelectedItem.ToString() == "Lithuania")
519     {
520         selectedNation = "Lithuania";
521     }
522     else if (picker.SelectedItem.ToString() == "Luxembourg")
523     {
524         selectedNation = "Luxembourg";
525     }
526     else if (picker.SelectedItem.ToString() == "Malta")
527     {
528         selectedNation = "Malta";
529     }
530     else if (picker.SelectedItem.ToString() == "Moldova")
531     {
532         selectedNation = "Moldova";
533     }
534     else if (picker.SelectedItem.ToString() == "Monaco")
535     {
```

```
536         selectedNation = "Monaco";
537     }
538     else if (picker.SelectedItem.ToString() == "Montenegro")
539     {
540         selectedNation = "Montenegro";
541     }
542     else if (picker.SelectedItem.ToString() == "Netherlands")
543     {
544         selectedNation = "Netherlands";
545     }
546     else if (picker.SelectedItem.ToString() == "North Macedonia")
547     {
548         selectedNation = "North Macedonia";
549     }
550     else if (picker.SelectedItem.ToString() == "Norway")
551     {
552         selectedNation = "Norway";
553     }
554     else if (picker.SelectedItem.ToString() == "Poland")
555     {
556         selectedNation = "Poland";
557     }
558     else if (picker.SelectedItem.ToString() == "Portugal")
559     {
560         selectedNation = "Portugal";
561     }
562     else if (picker.SelectedItem.ToString() == "Romania")
563     {
564         selectedNation = "Romania";
565     }
566     else if (picker.SelectedItem.ToString() == "Russia")
567     {
568         selectedNation = "Russia";
569     }
570     else if (picker.SelectedItem.ToString() == "San Marino")
571     {
572         selectedNation = "San Marino";
573     }
574     else if (picker.SelectedItem.ToString() == "Serbia")
575     {
576         selectedNation = "Serbia";
577     }
578     else if (picker.SelectedItem.ToString() == "Slovakia")
579     {
580         selectedNation = "Slovakia";
581     }
582     else if (picker.SelectedItem.ToString() == "Slovenia")
583     {
584         selectedNation = "Albania";
585     }
586     else if (picker.SelectedItem.ToString() == "Spain")
587     {
588         selectedNation = "Spain";
589     }
590     else if (picker.SelectedItem.ToString() == "Sweden")
591     {
592         selectedNation = "Sweden";
593     }
594     else if (picker.SelectedItem.ToString() == "Switzerland")
595     {
596         selectedNation = "Switzerland";
```



```
597     }
598     else if (picker.SelectedItem.ToString() == "Turkey")
599     {
600         selectedNation = "Turkey";
601     }
602     else if (picker.SelectedItem.ToString() == "Ukraine")
603     {
604         selectedNation = "Ukraine";
605     }
606     else if (picker.SelectedItem.ToString() == "United Kingdom")
607     {
608         selectedNation = "United Kingdom";
609     }
610     else if (picker.SelectedItem.ToString() == "Vatican City")
611     {
612         selectedNation = "Vatican City";
613     }
614
615     OnAppearing();
616 }
617
618 /** This function displays the next ten items on the leader board.
619 */
620 private void NextPageClicked(object sender, EventArgs e)
621 {
622     min = min + 10;
623     OnAppearing();
624 }
625
626 /** This function displays the first page of the leader board again.
627 */
628 private void FirstPageClicked(object sender, EventArgs e)
629 {
630     min = 0;
631     OnAppearing();
632 }
633 }
634 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3           xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5   x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.AdvancedPage"
6           xmlns:local="clr-namespace:Application_Green_Quake.Models">
7
8   <NavigationPage.TitleView>
9     <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10    VerticalOptions="EndAndExpand" Spacing="0">
11       <Label Text="Advanced" TextColor="White" FontSize="20" FontAttributes="Italic"
12    VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13       <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14    VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15     </StackLayout>
16   </NavigationPage.TitleView>
17
18   <ContentPage.Content>
19     <ScrollView>
20       <Grid Margin="5,5,5,5">
21         <Grid.RowDefinitions>
22           <RowDefinition Height="200" />
23         </Grid.RowDefinitions>
24         <Grid.ColumnDefinitions>
25           <ColumnDefinition />
26         </Grid.ColumnDefinitions>
27
28         <Image Grid.Column="0"
29           Grid.Row="0"
30           Aspect="AspectFill"
31           Source="{local:ImageResource
32    Application_Green_Quake.Images.SubCategories.Advanced.Fix.jpg}"/>
33         <StackLayout Grid.Column="0"
34           Grid.Row="0"
35           BackgroundColor="Black"
36           Opacity=".5">
37           <StackLayout.GestureRecognizers>
38             <TapGestureRecognizer Tapped="NavigateToFix"/>
39           </StackLayout.GestureRecognizers>
40         </StackLayout>
41         <StackLayout Grid.Column="0"
42           Grid.Row="0"
43           VerticalOptions="End"
44           Spacing="5"
45           Margin="40,0,40,15">
46           <StackLayout.GestureRecognizers>
47             <TapGestureRecognizer Tapped="NavigateToFix"/>
48           </StackLayout.GestureRecognizers>
49           <Label Text="Fix It"
50             TextColor="White"
51             HorizontalOptions="Center"
52             FontSize="24"
53             FontAttributes="Bold"
54             FontFamily="Proxima Nova"/>
55           <Label Text="Instead of throwing it away try fix it."
56             TextColor="White"
57             HorizontalOptions="Center"
58             FontSize="15"
59             FontFamily="Proxima Nova Thin"
60             HorizontalTextAlignment="Center"/>
61         </StackLayout>
62       </Grid>

```

```
58 |         </ScrollView>
59 |     </ContentPage.Content>
60 </ContentPage>
```

```
1 /!* \class The AdvancedPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the AdvancedPage View Class. This page displays and allows the
   navigation to each of the actions in the Advanced category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.AdvancedPageItems;
11 using System;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class AdvancedPage : ContentPage
19     {
20         public AdvancedPage()
21         {
22             InitializeComponent();
23             OnAppearing();
24         }
25
26         /** This function navigates to FixInsteadOfThrowAway.
27         */
28         private async void NavigateToFix(object sender, EventArgs e)
29         {
30             await Navigation.PushAsync(new FixInsteadOfThrowAway());
31         }
32         /** 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
33         * and display it in the navigation bar.
34         */
35         protected override void OnAppearing()
36         {
37             GetData data = new GetData();
38             data.SetLvl();
39
40             theLevel.Text = "LVL: " + GetData.lvl.ToString();
41         }
42     }
43 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5 x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.CommunityPage"
6             xmlns:local="clr-namespace:Application_Green_Quake.Models"
7             Title="Community Subcategories">
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Community" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15            Margin="0,0,30,0"/>
16        </StackLayout>
17    </NavigationPage.TitleView>
18
19    <ContentPage.Content>
20        <ScrollView>
21            <Grid Margin="5,5,5,5">
22                <Grid.RowDefinitions>
23                    <RowDefinition Height="200" />
24                    <RowDefinition Height="200" />
25                    <RowDefinition Height="200" />
26                    <RowDefinition Height="200" />
27                    <RowDefinition Height="200" />
28                    <RowDefinition Height="200" />
29                </Grid.RowDefinitions>
30                <Grid.ColumnDefinitions>
31                    <ColumnDefinition />
32                </Grid.ColumnDefinitions>
33
34                <Image Grid.Column="0"
35                    Grid.Row="0"
36                    Aspect="AspectFill"
37                    Source="{local:ImageResource
38        Application_Green_Quake.Images.SubCategories.Community.JoinE.jpg}"/>
39                <StackLayout Grid.Column="0"
40                    Grid.Row="0"
41                    BackgroundColor="Black"
42                    Opacity=".5">
43                    <StackLayout.GestureRecognizers>
44                        <TapGestureRecognizer Tapped="NavigateToEnvironmentalGroups"/>
45                    </StackLayout.GestureRecognizers>
46                </StackLayout>
47                <StackLayout Grid.Column="0"
48                    Grid.Row="0"
49                    VerticalOptions="End"
50                    Spacing="5"
51                    Margin="40,0,40,15">
52                    <StackLayout.GestureRecognizers>
53                        <TapGestureRecognizer Tapped="NavigateToEnvironmentalGroups"/>
54                    </StackLayout.GestureRecognizers>
55                    <Label Text="Join An Environmental Group"
56                        TextColor="White"
57                        HorizontalOptions="Center"
58                        FontSize="24"
59                        FontAttributes="Bold"
60                        FontFamily="Proxima Nova"
61                        HorizontalTextAlignment="Center"/>

```

```

57         <Label Text="Join an environmental group and do your part."
58             TextColor="White"
59             HorizontalOptions="Center"
60             FontSize="15"
61             FontFamily="Proxima Nova Thin"
62             HorizontalTextAlignment="Center"/>
63     </StackLayout>
64
65     <Image Grid.Column="0"
66         Grid.Row="1"
67         Aspect="AspectFill"
68         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.CreateE.jpg}"/>
69     <StackLayout Grid.Column="0"
70         Grid.Row="1"
71         BackgroundColor="Black"
72         Opacity=".5">
73         <StackLayout.GestureRecognizers>
74             <TapGestureRecognizer Tapped="NavigateToCreateEnvironmentalGroup"/>
75         </StackLayout.GestureRecognizers>
76     </StackLayout>
77     <StackLayout Grid.Column="0"
78         Grid.Row="1"
79         VerticalOptions="End"
80         Spacing="5"
81         Margin="40,0,40,15">
82         <StackLayout.GestureRecognizers>
83             <TapGestureRecognizer Tapped="NavigateToCreateEnvironmentalGroup"/>
84         </StackLayout.GestureRecognizers>
85         <Label Text="Create An Environmental Group"
86             TextColor="White"
87             HorizontalOptions="Center"
88             FontSize="24"
89             FontAttributes="Bold"
90             FontFamily="Proxima Nova"
91             HorizontalTextAlignment="Center"/>
92         <Label Text="No environmental group to join? No problem!"
93             TextColor="White"
94             HorizontalOptions="Center"
95             FontSize="15"
96             FontFamily="Proxima Nova Thin"
97             HorizontalTextAlignment="Center"/>
98     </StackLayout>
99
100    <Image Grid.Column="0"
101        Grid.Row="2"
102        Aspect="AspectFill"
103        Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.Spread.jpg}"/>
104    <StackLayout Grid.Column="0"
105        Grid.Row="2"
106        BackgroundColor="Black"
107        Opacity=".5">
108        <StackLayout.GestureRecognizers>
109            <TapGestureRecognizer Tapped="NavigateToSpreadAwareness"/>
110        </StackLayout.GestureRecognizers>
111    </StackLayout>
112    <StackLayout Grid.Column="0"
113        Grid.Row="2"
114        VerticalOptions="End"
115        Spacing="5"
116        Margin="40,0,40,15">

```

```
117         <StackLayout.GestureRecognizers>
118             <TapGestureRecognizer Tapped="NavigateToSpreadAwareness"/>
119         </StackLayout.GestureRecognizers>
120         <Label Text="Spread Awareness"
121             TextColor="White"
122             HorizontalOptions="Center"
123             FontSize="24"
124             FontAttributes="Bold"
125             FontFamily="Proxima Nova"/>
126         <Label Text="Spread awareness about environmental issues."
127             TextColor="White"
128             HorizontalOptions="Center"
129             FontSize="15"
130             FontFamily="Proxima Nova Thin"
131             HorizontalTextAlignment="Center"/>
132     </StackLayout>
133
134     <Image Grid.Column="0"
135         Grid.Row="3"
136         Aspect="AspectFill"
137         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.DoCommunity.jpg}"/>
138     <StackLayout Grid.Column="0"
139         Grid.Row="3"
140         BackgroundColor="Black"
141         Opacity=".5">
142         <StackLayout.GestureRecognizers>
143             <TapGestureRecognizer Tapped="NavigateToDoCommunity"/>
144         </StackLayout.GestureRecognizers>
145     </StackLayout>
146     <StackLayout Grid.Column="0"
147         Grid.Row="3"
148         VerticalOptions="End"
149         Spacing="5"
150         Margin="40,0,40,15">
151         <StackLayout.GestureRecognizers>
152             <TapGestureRecognizer Tapped="NavigateToDoCommunity"/>
153         </StackLayout.GestureRecognizers>
154         <Label Text="Do Something For The Community"
155             TextColor="White"
156             HorizontalOptions="Center"
157             HorizontalTextAlignment="Center"
158             FontSize="24"
159             FontAttributes="Bold"
160             FontFamily="Proxima Nova"/>
161         <Label Text="Do something for the community and improve peoples lives."
162             TextColor="White"
163             HorizontalOptions="Center"
164             FontSize="15"
165             FontFamily="Proxima Nova Thin"
166             HorizontalTextAlignment="Center"/>
167     </StackLayout>
168
169     <Image Grid.Column="0"
170         Grid.Row="4"
171         Aspect="AspectFill"
172         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.Donate.jpg}"/>
173     <StackLayout Grid.Column="0"
174         Grid.Row="4"
175         BackgroundColor="Black"
```

```

176         Opacity=".5">
177         <StackLayout.GestureRecognizers>
178             <TapGestureRecognizer Tapped="NavigateToDonateItems"/>
179         </StackLayout.GestureRecognizers>
180     </StackLayout>
181     <StackLayout Grid.Column="0"
182         Grid.Row="4"
183         VerticalOptions="End"
184         Spacing="5"
185         Margin="40,0,40,15">
186         <StackLayout.GestureRecognizers>
187             <TapGestureRecognizer Tapped="NavigateToDonateItems"/>
188         </StackLayout.GestureRecognizers>
189         <Label Text="Donate Something"
190             TextColor="White"
191             HorizontalOptions="Center"
192             FontSize="24"
193             FontAttributes="Bold"
194             FontFamily="Proxima Nova"/>
195         <Label Text="Donate an item instead of throwing it away or storing it
forever."
196             TextColor="White"
197             HorizontalOptions="Center"
198             FontSize="15"
199             FontFamily="Proxima Nova Thin"
200             HorizontalTextAlignment="Center"/>
201     </StackLayout>
202
203     <Image Grid.Column="0"
204         Grid.Row="5"
205         Aspect="AspectFill"
206         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Community.Share.jpg}"
207         Margin="0,0,0,5"/>
208     <StackLayout Grid.Column="0"
209         Grid.Row="5"
210         BackgroundColor="Black"
211         Opacity=".5"
212         Margin="0,0,0,5">
213         <StackLayout.GestureRecognizers>
214             <TapGestureRecognizer Tapped="NavigateToShareThisApp"/>
215         </StackLayout.GestureRecognizers>
216     </StackLayout>
217     <StackLayout Grid.Column="0"
218         Grid.Row="5"
219         VerticalOptions="End"
220         Spacing="5"
221         Margin="40,0,40,15">
222         <StackLayout.GestureRecognizers>
223             <TapGestureRecognizer Tapped="NavigateToShareThisApp"/>
224         </StackLayout.GestureRecognizers>
225         <Label Text="Share This App"
226             TextColor="White"
227             HorizontalOptions="Center"
228             FontSize="24"
229             FontAttributes="Bold"
230             FontFamily="Proxima Nova"/>
231         <Label Text="Share this app and lets grow together."
232             TextColor="White"
233             HorizontalOptions="Center"
234             FontSize="15"
235             FontFamily="Proxima Nova Thin"

```



```
236 |                                     HorizontalTextAlignment="Center"/>
237 |                                     </StackLayout>
238 |                                 </Grid>
239 |                             </ScrollView>
240 |                         </ContentPage.Content>
241 | </ContentPage>
```

```
1 /!* \class The CommunityPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the CommunityPage View Class. This page displays and allows the
   navigation to each of the actions in the Community category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Community;
11 using System;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class CommunityPage : ContentPage
19     {
20         public CommunityPage()
21         {
22             InitializeComponent();
23             OnAppearing();
24         }
25
26         /** This function navigates to EnvironmentalGroups.
27         */
28         private async void NavigateToEnvironmentalGroups(object sender, EventArgs e)
29         {
30             await Navigation.PushAsync(new EnvironmentalGroups());
31         }
32         /** This function navigates to CreateEnvironmentalGroup.
33         */
34         private async void NavigateToCreateEnvironmentalGroup(object sender, EventArgs e)
35         {
36             await Navigation.PushAsync(new CreateEnvironmentalGroup());
37         }
38         /** This function navigates to SpreadAwareness.
39         */
40         private async void NavigateToSpreadAwareness(object sender, EventArgs e)
41         {
42             await Navigation.PushAsync(new SpreadAwareness());
43         }
44         /** This function navigates to DoCommunity.
45         */
46         private async void NavigateToDoCommunity(object sender, EventArgs e)
47         {
48             await Navigation.PushAsync(new DoCommunity());
49         }
50         /** This function navigates to DonateItems.
51         */
52         private async void NavigateToDonateItems(object sender, EventArgs e)
53         {
54             await Navigation.PushAsync(new DonateItems());
55         }
56         /** This function navigates to ShareThisApp.
57         */
58         private async void NavigateToShareThisApp(object sender, EventArgs e)
59         {
```

```
60         await Navigation.PushAsync(new ShareThisApp());
61     }
62     /** 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
63     * and display it in the navigation bar.
64     */
65     protected override void OnAppearing()
66     {
67         GetData data = new GetData();
68         data.SetLvl();
69
70         theLevel.Text = "LVL: " + GetData.lvl.ToString();
71     }
72 }
73 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3           xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5 x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.EnergyPage"
6           xmlns:local="clr-namespace:Application_Green_Quake.Models"
7           Title="Energy Subcategories">
8     <NavigationPage.TitleView>
9       <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10      VerticalOptions="EndAndExpand" Spacing="0">
11         <Label Text="Energy" TextColor="White" FontSize="20" FontAttributes="Italic"
12      VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13         <Label x:Name="theLevel" TextColor="White" FontSize="20"
14      FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15      Margin="0,0,30,0"/>
16       </StackLayout>
17     </NavigationPage.TitleView>
18
19     <ContentPage.Content>
20       <ScrollView>
21         <Grid Margin="5,5,5,5">
22           <Grid.RowDefinitions>
23             <RowDefinition Height="200" />
24             <RowDefinition Height="200" />
25             <RowDefinition Height="200" />
26             <RowDefinition Height="200" />
27             <RowDefinition Height="200" />
28             <RowDefinition Height="200" />
29             <RowDefinition Height="200" />
30             <RowDefinition Height="200" />
31             <RowDefinition Height="200" />
32             <RowDefinition Height="200" />
33             <RowDefinition Height="200" />
34             <RowDefinition Height="200" />
35           </Grid.RowDefinitions>
36           <Grid.ColumnDefinitions>
37             <ColumnDefinition />
38           </Grid.ColumnDefinitions>
39
40           <Image Grid.Column="0"
41             Grid.Row="0"
42             Aspect="AspectFill"
43             Source="{local:ImageResource
44      Application_Green_Quake.Images.SubCategories.Energy.FullDryer.jpg}"/>
45           <StackLayout Grid.Column="0"
46             Grid.Row="0"
47             BackgroundColor="Black"
48             Opacity=".5">
49             <StackLayout.GestureRecognizers>
50               <TapGestureRecognizer Tapped="NavigateToDryerFull"/>
51             </StackLayout.GestureRecognizers>
52           </StackLayout>
53           <StackLayout Grid.Column="0"
54             Grid.Row="0"
55             VerticalOptions="End"
56             Spacing="5"
57             Margin="40,0,40,15">

```

```
57         <StackLayout.GestureRecognizers>
58             <TapGestureRecognizer Tapped="NavigateToDryerFull"/>
59         </StackLayout.GestureRecognizers>
60         <Label Text="Full Dryer"
61             TextColor="White"
62             HorizontalOptions="Center"
63             FontSize="24"
64             FontAttributes="Bold"
65             FontFamily="Proxima Nova"/>
66         <Label Text="Only use the Dryer when it is full."
67             TextColor="White"
68             HorizontalOptions="Center"
69             FontSize="15"
70             FontFamily="Proxima Nova Thin"
71             HorizontalTextAlignment="Center"/>
72     </StackLayout>
73
74     <Image Grid.Column="0"
75         Grid.Row="1"
76         Aspect="AspectFill"
77         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.FullWashingMachine.jpg}"/>
78     <StackLayout Grid.Column="0"
79         Grid.Row="1"
80         BackgroundColor="Black"
81         Opacity=".5">
82         <StackLayout.GestureRecognizers>
83             <TapGestureRecognizer Tapped="NavigateToMachineFull"/>
84         </StackLayout.GestureRecognizers>
85     </StackLayout>
86     <StackLayout Grid.Column="0"
87         Grid.Row="1"
88         VerticalOptions="End"
89         Spacing="5"
90         Margin="40,0,40,15">
91         <StackLayout.GestureRecognizers>
92             <TapGestureRecognizer Tapped="NavigateToMachineFull"/>
93         </StackLayout.GestureRecognizers>
94         <Label Text="Full Washing Machine"
95             TextColor="White"
96             HorizontalOptions="Center"
97             FontSize="24"
98             FontAttributes="Bold"
99             FontFamily="Proxima Nova"/>
100        <Label Text="Only use the washing machine when it is full."
101            TextColor="White"
102            HorizontalOptions="Center"
103            FontSize="15"
104            FontFamily="Proxima Nova Thin"
105            HorizontalTextAlignment="Center"/>
106    </StackLayout>
107
108    <Image Grid.Column="0"
109        Grid.Row="2"
110        Aspect="AspectFill"
111        Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.FullDishwasher.jpg}"/>
112    <StackLayout Grid.Column="0"
113        Grid.Row="2"
114        BackgroundColor="Black"
115        Opacity=".5">
116        <StackLayout.GestureRecognizers>
```

```

117         <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
118     </StackLayout.GestureRecognizers>
119 </StackLayout>
120 <StackLayout Grid.Column="0"
121     Grid.Row="2"
122     VerticalOptions="End"
123     Spacing="5"
124     Margin="40,0,40,15">
125     <StackLayout.GestureRecognizers>
126         <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
127     </StackLayout.GestureRecognizers>
128     <Label Text="Full Dishwasher"
129         TextColor="White"
130         HorizontalOptions="Center"
131         FontSize="24"
132         FontAttributes="Bold"
133         FontFamily="Proxima Nova"/>
134     <Label Text="Only use the dishwasher when it is full."
135         TextColor="White"
136         HorizontalOptions="Center"
137         FontSize="15"
138         FontFamily="Proxima Nova Thin"
139         HorizontalTextAlignment="Center"/>
140 </StackLayout>
141
142 <Image Grid.Column="0"
143     Grid.Row="3"
144     Aspect="AspectFill"
145     Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.HangDry.jpg}"/>
146 <StackLayout Grid.Column="0"
147     Grid.Row="3"
148     BackgroundColor="Black"
149     Opacity=".5">
150     <StackLayout.GestureRecognizers>
151         <TapGestureRecognizer Tapped="NavigateToHangDry"/>
152     </StackLayout.GestureRecognizers>
153 </StackLayout>
154 <StackLayout Grid.Column="0"
155     Grid.Row="3"
156     VerticalOptions="End"
157     Spacing="5"
158     Margin="40,0,40,15">
159     <StackLayout.GestureRecognizers>
160         <TapGestureRecognizer Tapped="NavigateToHangDry"/>
161     </StackLayout.GestureRecognizers>
162     <Label Text="Hang Dry Clothes"
163         TextColor="White"
164         HorizontalOptions="Center"
165         FontSize="24"
166         FontAttributes="Bold"
167         FontFamily="Proxima Nova"/>
168     <Label Text="Instead of using the washing machine. Hang dry your
clothes instead."
169         TextColor="White"
170         HorizontalOptions="Center"
171         FontSize="15"
172         FontFamily="Proxima Nova Thin"
173         HorizontalTextAlignment="Center"/>
174 </StackLayout>
175

```

```
176         <Image Grid.Column="0"
177             Grid.Row="4"
178             Aspect="AspectFill"
179             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.Thermostat.jpg}"/>
180         <StackLayout Grid.Column="0"
181             Grid.Row="4"
182             BackgroundColor="Black"
183             Opacity=".5">
184             <StackLayout.GestureRecognizers>
185                 <TapGestureRecognizer Tapped="NavigateToEfficientThermostat"/>
186             </StackLayout.GestureRecognizers>
187         </StackLayout>
188         <StackLayout Grid.Column="0"
189             Grid.Row="4"
190             VerticalOptions="End"
191             Spacing="5"
192             Margin="40,0,40,15">
193             <StackLayout.GestureRecognizers>
194                 <TapGestureRecognizer Tapped="NavigateToEfficientThermostat"/>
195             </StackLayout.GestureRecognizers>
196             <Label Text="Efficient Thermostat"
197                 TextColor="White"
198                 HorizontalOptions="Center"
199                 FontSize="24"
200                 FontAttributes="Bold"
201                 FontFamily="Proxima Nova"/>
202             <Label Text="Efficiently program your thermostat."
203                 TextColor="White"
204                 HorizontalOptions="Center"
205                 FontSize="15"
206                 FontFamily="Proxima Nova Thin"
207                 HorizontalTextAlignment="Center"/>
208         </StackLayout>
209
210         <Image Grid.Column="0"
211             Grid.Row="5"
212             Aspect="AspectFill"
213             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.SolarPanel.jpg}"/>
214         <StackLayout Grid.Column="0"
215             Grid.Row="5"
216             BackgroundColor="Black"
217             Opacity=".5">
218             <StackLayout.GestureRecognizers>
219                 <TapGestureRecognizer Tapped="NavigateToSolarPanel"/>
220             </StackLayout.GestureRecognizers>
221         </StackLayout>
222         <StackLayout Grid.Column="0"
223             Grid.Row="5"
224             VerticalOptions="End"
225             Spacing="5"
226             Margin="40,0,40,15">
227             <StackLayout.GestureRecognizers>
228                 <TapGestureRecognizer Tapped="NavigateToSolarPanel"/>
229             </StackLayout.GestureRecognizers>
230             <Label Text="Solar Energy"
231                 TextColor="White"
232                 HorizontalOptions="Center"
233                 FontSize="24"
234                 FontAttributes="Bold"
235                 FontFamily="Proxima Nova"/>
```

```
236         <Label Text="Install a solar panel on your home or garden."
237             TextColor="White"
238             HorizontalOptions="Center"
239             FontSize="15"
240             FontFamily="Proxima Nova Thin"
241             HorizontalTextAlignment="Center"/>
242     </StackLayout>
243
244     <Image Grid.Column="0"
245         Grid.Row="6"
246         Aspect="AspectFill"
247         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.LedLight.jpg}"/>
248     <StackLayout Grid.Column="0"
249         Grid.Row="6"
250         BackgroundColor="Black"
251         Opacity=".5">
252         <StackLayout.GestureRecognizers>
253             <TapGestureRecognizer Tapped="NavigateToLedLightbulb"/>
254         </StackLayout.GestureRecognizers>
255     </StackLayout>
256     <StackLayout Grid.Column="0"
257         Grid.Row="6"
258         VerticalOptions="End"
259         Spacing="5"
260         Margin="40,0,40,15">
261         <StackLayout.GestureRecognizers>
262             <TapGestureRecognizer Tapped="NavigateToLedLightbulb"/>
263         </StackLayout.GestureRecognizers>
264         <Label Text="Led Lights"
265             TextColor="White"
266             HorizontalOptions="Center"
267             FontSize="24"
268             FontAttributes="Bold"
269             FontFamily="Proxima Nova"/>
270         <Label Text="Install Led Light Bulbs instead of normal ones."
271             TextColor="White"
272             HorizontalOptions="Center"
273             FontSize="15"
274             FontFamily="Proxima Nova Thin"
275             HorizontalTextAlignment="Center"/>
276     </StackLayout>
277
278     <Image Grid.Column="0"
279         Grid.Row="7"
280         Aspect="AspectFill"
281         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.OffSocket.jpg}"/>
282     <StackLayout Grid.Column="0"
283         Grid.Row="7"
284         BackgroundColor="Black"
285         Opacity=".5">
286         <StackLayout.GestureRecognizers>
287             <TapGestureRecognizer Tapped="NavigateToOffSocketSwitch"/>
288         </StackLayout.GestureRecognizers>
289     </StackLayout>
290     <StackLayout Grid.Column="0"
291         Grid.Row="7"
292         VerticalOptions="End"
293         Spacing="5"
294         Margin="40,0,40,15">
```



```

295         <StackLayout.GestureRecognizers>
296             <TapGestureRecognizer Tapped="NavigateToOffSocketSwitch"/>
297         </StackLayout.GestureRecognizers>
298         <Label Text="Socket Off"
299             TextColor="White"
300             HorizontalOptions="Center"
301             FontSize="24"
302             FontAttributes="Bold"
303             FontFamily="Proxima Nova"/>
304         <Label Text="Turn off or unplug devices not in use from the wall
socket."
305             TextColor="White"
306             HorizontalOptions="Center"
307             FontSize="15"
308             FontFamily="Proxima Nova Thin"
309             HorizontalTextAlignment="Center"/>
310     </StackLayout>
311
312     <Image Grid.Column="0"
313         Grid.Row="8"
314         Aspect="AspectFill"
315         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.OffLights.jpg}"/>
316     <StackLayout Grid.Column="0"
317         Grid.Row="8"
318         BackgroundColor="Black"
319         Opacity=".5">
320         <StackLayout.GestureRecognizers>
321             <TapGestureRecognizer Tapped="NavigateToOffLights"/>
322         </StackLayout.GestureRecognizers>
323     </StackLayout>
324     <StackLayout Grid.Column="0"
325         Grid.Row="8"
326         VerticalOptions="End"
327         Spacing="5"
328         Margin="40,0,40,15">
329         <StackLayout.GestureRecognizers>
330             <TapGestureRecognizer Tapped="NavigateToOffLights"/>
331         </StackLayout.GestureRecognizers>
332         <Label Text="Lights Off"
333             TextColor="White"
334             HorizontalOptions="Center"
335             FontSize="24"
336             FontAttributes="Bold"
337             FontFamily="Proxima Nova"/>
338         <Label Text="Turn off the lights when leaving the room."
339             TextColor="White"
340             HorizontalOptions="Center"
341             FontSize="15"
342             FontFamily="Proxima Nova Thin"
343             HorizontalTextAlignment="Center"/>
344     </StackLayout>
345
346     <Image Grid.Column="0"
347         Grid.Row="9"
348         Aspect="AspectFill"
349         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.InsulateHome.jpg}"/>
350     <StackLayout Grid.Column="0"
351         Grid.Row="9"
352         BackgroundColor="Black"
353         Opacity=".5">

```

```

354         <StackLayout.GestureRecognizers>
355             <TapGestureRecognizer Tapped="NavigateToIsolateHome"/>
356         </StackLayout.GestureRecognizers>
357     </StackLayout>
358     <StackLayout Grid.Column="0"
359         Grid.Row="9"
360         VerticalOptions="End"
361         Spacing="5"
362         Margin="40,0,40,15">
363         <StackLayout.GestureRecognizers>
364             <TapGestureRecognizer Tapped="NavigateToIsolateHome"/>
365         </StackLayout.GestureRecognizers>
366         <Label Text="Insulate The Home"
367             TextColor="White"
368             HorizontalOptions="Center"
369             FontSize="24"
370             FontAttributes="Bold"
371             FontFamily="Proxima Nova"/>
372         <Label Text="Insulate your home and stay warm."
373             TextColor="White"
374             HorizontalOptions="Center"
375             FontSize="15"
376             FontFamily="Proxima Nova Thin"
377             HorizontalTextAlignment="Center"/>
378     </StackLayout>
379
380     <Image Grid.Column="0"
381         Grid.Row="10"
382         Aspect="AspectFill"
383         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.Microwave.jpg}"/>
384     <StackLayout Grid.Column="0"
385         Grid.Row="10"
386         BackgroundColor="Black"
387         Opacity=".5">
388         <StackLayout.GestureRecognizers>
389             <TapGestureRecognizer Tapped="NavigateToMicrowaveNotOven"/>
390         </StackLayout.GestureRecognizers>
391     </StackLayout>
392     <StackLayout Grid.Column="0"
393         Grid.Row="10"
394         VerticalOptions="End"
395         Spacing="5"
396         Margin="40,0,40,15">
397         <StackLayout.GestureRecognizers>
398             <TapGestureRecognizer Tapped="NavigateToMicrowaveNotOven"/>
399         </StackLayout.GestureRecognizers>
400         <Label Text="Microwave Don't Oven"
401             TextColor="White"
402             HorizontalOptions="Center"
403             FontSize="24"
404             FontAttributes="Bold"
405             FontFamily="Proxima Nova"/>
406         <Label Text="When possible use the microwave over the oven."
407             TextColor="White"
408             HorizontalOptions="Center"
409             FontSize="15"
410             FontFamily="Proxima Nova Thin"
411             HorizontalTextAlignment="Center"/>
412     </StackLayout>
413

```

```
414         <Image Grid.Column="0"
415             Grid.Row="11"
416             Aspect="AspectFill"
417             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.ReBatteries.jpg}"/>
418         <StackLayout Grid.Column="0"
419             Grid.Row="11"
420             BackgroundColor="Black"
421             Opacity=".5">
422             <StackLayout.GestureRecognizers>
423                 <TapGestureRecognizer Tapped="NavigateToReBatteries"/>
424             </StackLayout.GestureRecognizers>
425         </StackLayout>
426         <StackLayout Grid.Column="0"
427             Grid.Row="11"
428             VerticalOptions="End"
429             Spacing="5"
430             Margin="40,0,40,15">
431             <StackLayout.GestureRecognizers>
432                 <TapGestureRecognizer Tapped="NavigateToReBatteries"/>
433             </StackLayout.GestureRecognizers>
434             <Label Text="Reusable Batteries"
435                 TextColor="White"
436                 HorizontalOptions="Center"
437                 FontSize="24"
438                 FontAttributes="Bold"
439                 FontFamily="Proxima Nova"/>
440             <Label Text="Use Rechargeable Batteries over non rechargeable ones."
441                 TextColor="White"
442                 HorizontalOptions="Center"
443                 FontSize="15"
444                 FontFamily="Proxima Nova Thin"
445                 HorizontalTextAlignment="Center"/>
446         </StackLayout>
447
448         <Image Grid.Column="0"
449             Grid.Row="12"
450             Aspect="AspectFill"
451             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.TurnDownFridge.jpg}"/>
452         <StackLayout Grid.Column="0"
453             Grid.Row="12"
454             BackgroundColor="Black"
455             Opacity=".5">
456             <StackLayout.GestureRecognizers>
457                 <TapGestureRecognizer Tapped="NavigateToRefrigiratorDown"/>
458             </StackLayout.GestureRecognizers>
459         </StackLayout>
460         <StackLayout Grid.Column="0"
461             Grid.Row="12"
462             VerticalOptions="End"
463             Spacing="5"
464             Margin="40,0,40,15">
465             <StackLayout.GestureRecognizers>
466                 <TapGestureRecognizer Tapped="NavigateToRefrigiratorDown"/>
467             </StackLayout.GestureRecognizers>
468             <Label Text="Turn Down The Fridge"
469                 TextColor="White"
470                 HorizontalOptions="Center"
471                 FontSize="24"
472                 FontAttributes="Bold"
473                 FontFamily="Proxima Nova"/>
```

```
474         <Label Text="Turn down the refrigerator to use less energy."
475             TextColor="White"
476             HorizontalOptions="Center"
477             FontSize="15"
478             FontFamily="Proxima Nova Thin"
479             HorizontalTextAlignment="Center"/>
480     </StackLayout>
481
482     <Image Grid.Column="0"
483         Grid.Row="13"
484         Aspect="AspectFill"
485         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.InsulateWaterTank.jpg}"/>
486     <StackLayout Grid.Column="0"
487         Grid.Row="13"
488         BackgroundColor="Black"
489         Opacity=".5">
490         <StackLayout.GestureRecognizers>
491             <TapGestureRecognizer Tapped="NavigateToInsulateWater"/>
492         </StackLayout.GestureRecognizers>
493     </StackLayout>
494     <StackLayout Grid.Column="0"
495         Grid.Row="13"
496         VerticalOptions="End"
497         Spacing="5"
498         Margin="40,0,40,15">
499         <StackLayout.GestureRecognizers>
500             <TapGestureRecognizer Tapped="NavigateToInsulateWater"/>
501         </StackLayout.GestureRecognizers>
502         <Label Text="Insulate The Water Tank"
503             TextColor="White"
504             HorizontalOptions="Center"
505             FontSize="24"
506             FontAttributes="Bold"
507             FontFamily="Proxima Nova"/>
508         <Label Text="Insulate the water tank and keep the water warm."
509             TextColor="White"
510             HorizontalOptions="Center"
511             FontSize="15"
512             FontFamily="Proxima Nova Thin"
513             HorizontalTextAlignment="Center"/>
514     </StackLayout>
515
516     <Image Grid.Column="0"
517         Grid.Row="14"
518         Aspect="AspectFill"
519         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.SealDraft.jpg}"/>
520     <StackLayout Grid.Column="0"
521         Grid.Row="14"
522         BackgroundColor="Black"
523         Opacity=".5">
524         <StackLayout.GestureRecognizers>
525             <TapGestureRecognizer Tapped="NavigateToSealDrafts"/>
526         </StackLayout.GestureRecognizers>
527     </StackLayout>
528     <StackLayout Grid.Column="0"
529         Grid.Row="14"
530         VerticalOptions="End"
531         Spacing="5"
532         Margin="40,0,40,15">
```

```
533         <StackLayout.GestureRecognizers>
534             <TapGestureRecognizer Tapped="NavigateToSealDrafts"/>
535         </StackLayout.GestureRecognizers>
536         <Label Text="Seal The Drafts"
537             TextColor="White"
538             HorizontalOptions="Center"
539             FontSize="24"
540             FontAttributes="Bold"
541             FontFamily="Proxima Nova"/>
542         <Label Text="Seal the drafts in your home and keep the cold outside."
543             TextColor="White"
544             HorizontalOptions="Center"
545             FontSize="15"
546             FontFamily="Proxima Nova Thin"
547             HorizontalTextAlignment="Center"/>
548     </StackLayout>
549
550     <Image Grid.Column="0"
551         Grid.Row="15"
552         Aspect="AspectFill"
553         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.SealDuct.jpg}"
554         Margin="0,0,0,5"/>
555     <StackLayout Grid.Column="0"
556         Grid.Row="15"
557         BackgroundColor="Black"
558         Opacity=".5"
559         Margin="0,0,0,5">
560         <StackLayout.GestureRecognizers>
561             <TapGestureRecognizer Tapped="NavigateToSealDucts"/>
562         </StackLayout.GestureRecognizers>
563     </StackLayout>
564     <StackLayout Grid.Column="0"
565         Grid.Row="15"
566         VerticalOptions="End"
567         Spacing="5"
568         Margin="40,0,40,15">
569         <StackLayout.GestureRecognizers>
570             <TapGestureRecognizer Tapped="NavigateToSealDucts"/>
571         </StackLayout.GestureRecognizers>
572         <Label Text="Seal The Ducts"
573             TextColor="White"
574             HorizontalOptions="Center"
575             FontSize="24"
576             FontAttributes="Bold"
577             FontFamily="Proxima Nova"/>
578         <Label Text="Seal the ducts in your home and keep the cold out."
579             TextColor="White"
580             HorizontalOptions="Center"
581             FontSize="15"
582             FontFamily="Proxima Nova Thin"
583             HorizontalTextAlignment="Center"/>
584     </StackLayout>
585 </Grid>
586 </ScrollView>
587 </ContentPage.Content>
588 </ContentPage>
```

```
1 /!* \class The EnergyPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the EnergyPage View Class. This page displays and allows the
   navigation to each of the actions in the Energy category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Energy;
11 using Application_Green_Quake.Views.EcoActions.Habits;
12 using System;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EnergyPage : ContentPage
20     {
21         public EnergyPage()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function navigates to DryerFull.
27         */
28         private async void NavigateToDryerFull(object sender, EventArgs e)
29         {
30             await Navigation.PushAsync(new DryerFull());
31         }
32         /** This function navigates to MachineFull.
33         */
34         private async void NavigateToMachineFull(object sender, EventArgs e)
35         {
36             await Navigation.PushAsync(new MachineFull());
37         }
38         /** This function navigates to DishwasherFull.
39         */
40         private async void NavigateToDishwasherFull(object sender, EventArgs e)
41         {
42             await Navigation.PushAsync(new DishwasherFull());
43         }
44         /** This function navigates to HangDry.
45         */
46         private async void NavigateToHangDry(object sender, EventArgs e)
47         {
48             await Navigation.PushAsync(new HangDry());
49         }
50         /** This function navigates to EfficientThermostat.
51         */
52         private async void NavigateToEfficientThermostat(object sender, EventArgs e)
53         {
54             await Navigation.PushAsync(new EfficientThermostat());
55         }
56         /** This function navigates to SolarPanel.
57         */
58         private async void NavigateToSolarPanel(object sender, EventArgs e)
59         {
60             await Navigation.PushAsync(new SolarPanel());
61         }
62     }
63 }
```

```
60     private async void NavigateToSolarPanel(object sender, EventArgs e)
61     {
62         await Navigation.PushAsync(new SolarPanel());
63     }
64     /** This function navigates to LedLightBulb.
65     */
66     private async void NavigateToLedLightbulb(object sender, EventArgs e)
67     {
68         await Navigation.PushAsync(new LedLightBulb());
69     }
70     /** This function navigates to OffSocketSwitch.
71     */
72     private async void NavigateToOffSocketSwitch(object sender, EventArgs e)
73     {
74         await Navigation.PushAsync(new OffSocketSwitch());
75     }
76     /** This function navigates to TurnOffLights.
77     */
78     private async void NavigateToOffLights(object sender, EventArgs e)
79     {
80         await Navigation.PushAsync(new TurnOffLights());
81     }
82     /** This function navigates to IsolateHome.
83     */
84     private async void NavigateToIsolateHome(object sender, EventArgs e)
85     {
86         await Navigation.PushAsync(new IsolateHome());
87     }
88     /** This function navigates to MicrowaveNotOven.
89     */
90     private async void NavigateToMicrowaveNotOven(object sender, EventArgs e)
91     {
92         await Navigation.PushAsync(new MicrowaveNotOven());
93     }
94     /** This function navigates to ReBatteries.
95     */
96     private async void NavigateToReBatteries(object sender, EventArgs e)
97     {
98         await Navigation.PushAsync(new ReBatteries());
99     }
100    /** This function navigates to RefrigeratorDown.
101    */
102    private async void NavigateToRefrigeratorDown(object sender, EventArgs e)
103    {
104        await Navigation.PushAsync(new RefrigeratorDown());
105    }
106    /** This function navigates to InsulateWater.
107    */
108    private async void NavigateToInsulateWater(object sender, EventArgs e)
109    {
110        await Navigation.PushAsync(new InsulateWater());
111    }
112    /** This function navigates to SealDrafts.
113    */
114    private async void NavigateToSealDrafts(object sender, EventArgs e)
115    {
116        await Navigation.PushAsync(new SealDrafts());
117    }
118    /** This function navigates to SealDucts.
119    */
120    private async void NavigateToSealDucts(object sender, EventArgs e)
```

```
121     {
122         await Navigation.PushAsync(new SealDucts());
123     }
124     /** 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
125     * and display it in the navigation bar.
126     */
127     protected override void OnAppearing()
128     {
129         GetData data = new GetData();
130         data.SetLvl();
131
132         theLevel.Text = "LVL: " + GetData.lvl.ToString();
133     }
134 }
135 }
```



```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.FoodAndDrinkPage"
6     xmlns:local="clr-namespace:Application_Green_Quake.Models"
7     Title="Food and Drink Subcategories">
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Food and Drink" TextColor="White" FontSize="20"
12            FontAttributes="Italic" VerticalOptions="CenterAndExpand"
13            HorizontalOptions="StartAndExpand"/>
14             <Label x:Name="theLevel" TextColor="White" FontSize="20"
15            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
16            Margin="0,0,30,0"/>
17        </StackLayout>
18    </NavigationPage.TitleView>
19
20    <ContentPage.Content>
21        <ScrollView>
22            <Grid Margin="5,5,5,5">
23                <Grid.RowDefinitions>
24                    <RowDefinition Height="200" />
25                    <RowDefinition Height="200" />
26                    <RowDefinition Height="200" />
27                    <RowDefinition Height="200" />
28                    <RowDefinition Height="200" />
29                    <RowDefinition Height="200" />
30                    <RowDefinition Height="200" />
31                    <RowDefinition Height="200" />
32                    <RowDefinition Height="200" />
33                </Grid.RowDefinitions>
34                <Grid.ColumnDefinitions>
35                    <ColumnDefinition />
36                </Grid.ColumnDefinitions>
37
38                <Image Grid.Column="0"
39                    Grid.Row="0"
40                    Aspect="AspectFill"
41                    Source="{local:ImageResource
42        Application_Green_Quake.Images.SubCategories.FD.OrganicFood.jpg}"/>
43                <StackLayout Grid.Column="0"
44                    Grid.Row="0"
45                    BackgroundColor="Black"
46                    Opacity=".5">
47                    <StackLayout.GestureRecognizers>
48                        <TapGestureRecognizer Tapped="NavigateToBuyOrganicFood"/>
49                    </StackLayout.GestureRecognizers>
50                </StackLayout>
51                <StackLayout Grid.Column="0"
52                    Grid.Row="0"
53                    VerticalOptions="End"
54                    Spacing="5"
55                    Margin="40,0,40,15">
56                    <StackLayout.GestureRecognizers>
57                        <TapGestureRecognizer Tapped="NavigateToBuyOrganicFood"/>
58                    </StackLayout.GestureRecognizers>
59                    <Label Text="Go Organic"
60                        TextColor="White"

```

```

56         HorizontalOptions="Center"
57         FontSize="24"
58         FontAttributes="Bold"
59         FontFamily="Proxima Nova"/>
60     <Label Text="Purchase or make organic food over non organic."
61         TextColor="White"
62         HorizontalOptions="Center"
63         FontSize="15"
64         FontFamily="Proxima Nova Thin"
65         HorizontalTextAlignment="Center"/>
66 </StackLayout>
67
68 <Image Grid.Column="0"
69         Grid.Row="1"
70         Aspect="AspectFill"
71         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.WaterOverFiz.jpg}"/>
72 <StackLayout Grid.Column="0"
73         Grid.Row="1"
74         BackgroundColor="Black"
75         Opacity=".5">
76     <StackLayout.GestureRecognizers>
77         <TapGestureRecognizer Tapped="NavigateToWaterOverFizzy"/>
78     </StackLayout.GestureRecognizers>
79 </StackLayout>
80 <StackLayout Grid.Column="0"
81         Grid.Row="1"
82         VerticalOptions="End"
83         Spacing="5"
84         Margin="40,0,40,15">
85     <StackLayout.GestureRecognizers>
86         <TapGestureRecognizer Tapped="NavigateToWaterOverFizzy"/>
87     </StackLayout.GestureRecognizers>
88     <Label Text="Keeping It H2O"
89         TextColor="White"
90         HorizontalOptions="Center"
91         FontSize="24"
92         FontAttributes="Bold"
93         FontFamily="Proxima Nova"/>
94     <Label Text="Purchase or have water over other drinks."
95         TextColor="White"
96         HorizontalOptions="Center"
97         FontSize="15"
98         FontFamily="Proxima Nova Thin"
99         HorizontalTextAlignment="Center"/>
100 </StackLayout>
101
102 <Image Grid.Column="0"
103         Grid.Row="2"
104         Aspect="AspectFill"
105         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.EatAllYouMake.jpg}"/>
106 <StackLayout Grid.Column="0"
107         Grid.Row="2"
108         BackgroundColor="Black"
109         Opacity=".5">
110     <StackLayout.GestureRecognizers>
111         <TapGestureRecognizer Tapped="NavigateToEatAllYouMake"/>
112     </StackLayout.GestureRecognizers>
113 </StackLayout>
114 <StackLayout Grid.Column="0"
115         Grid.Row="2"

```

```

116         VerticalOptions="End"
117         Spacing="5"
118         Margin="40,0,40,15">
119     <StackLayout.GestureRecognizers>
120         <TapGestureRecognizer Tapped="NavigateToEatAllYouMake"/>
121     </StackLayout.GestureRecognizers>
122     <Label Text="Eat Everything"
123         TextColor="White"
124         HorizontalOptions="Center"
125         FontSize="24"
126         FontAttributes="Bold"
127         FontFamily="Proxima Nova"/>
128     <Label Text="Make just enough food so you can eat it all."
129         TextColor="White"
130         HorizontalOptions="Center"
131         FontSize="15"
132         FontFamily="Proxima Nova Thin"
133         HorizontalTextAlignment="Center"/>
134 </StackLayout>
135
136     <Image Grid.Column="0"
137         Grid.Row="3"
138         Aspect="AspectFill"
139         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.SaveLeftovers.jpg}"/>
140     <StackLayout Grid.Column="0"
141         Grid.Row="3"
142         BackgroundColor="Black"
143         Opacity=".5">
144         <StackLayout.GestureRecognizers>
145             <TapGestureRecognizer Tapped="NavigateToSaveLeftovers"/>
146         </StackLayout.GestureRecognizers>
147     </StackLayout>
148     <StackLayout Grid.Column="0"
149         Grid.Row="3"
150         VerticalOptions="End"
151         Spacing="5"
152         Margin="40,0,40,15">
153     <StackLayout.GestureRecognizers>
154         <TapGestureRecognizer Tapped="NavigateToSaveLeftovers"/>
155     </StackLayout.GestureRecognizers>
156     <Label Text="Save Leftovers"
157         TextColor="White"
158         HorizontalOptions="Center"
159         FontSize="24"
160         FontAttributes="Bold"
161         FontFamily="Proxima Nova"/>
162     <Label Text="Save the leftovers for another time."
163         TextColor="White"
164         HorizontalOptions="Center"
165         FontSize="15"
166         FontFamily="Proxima Nova Thin"
167         HorizontalTextAlignment="Center"/>
168 </StackLayout>
169
170     <Image Grid.Column="0"
171         Grid.Row="4"
172         Aspect="AspectFill"
173         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.NoMeat.jpg}"/>
174     <StackLayout Grid.Column="0"

```

```

175         Grid.Row="4"
176         BackgroundColor="Black"
177         Opacity=".5">
178     <StackLayout.GestureRecognizers>
179         <TapGestureRecognizer Tapped="NavigateToNoMeat"/>
180     </StackLayout.GestureRecognizers>
181 </StackLayout>
182 <StackLayout Grid.Column="0"
183     Grid.Row="4"
184     VerticalOptions="End"
185     Spacing="5"
186     Margin="40,0,40,15">
187     <StackLayout.GestureRecognizers>
188         <TapGestureRecognizer Tapped="NavigateToNoMeat"/>
189     </StackLayout.GestureRecognizers>
190     <Label Text="No Meat"
191         TextColor="White"
192         HorizontalOptions="Center"
193         FontSize="24"
194         FontAttributes="Bold"
195         FontFamily="Proxima Nova"/>
196     <Label Text="Eat something else over meat today."
197         TextColor="White"
198         HorizontalOptions="Center"
199         FontSize="15"
200         FontFamily="Proxima Nova Thin"
201         HorizontalTextAlignment="Center"/>
202 </StackLayout>
203
204     <Image Grid.Column="0"
205         Grid.Row="5"
206         Aspect="AspectFill"
207         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.ReCoffeeMug.jpg}"/>
208     <StackLayout Grid.Column="0"
209         Grid.Row="5"
210         BackgroundColor="Black"
211         Opacity=".5">
212     <StackLayout.GestureRecognizers>
213         <TapGestureRecognizer Tapped="NavigateToReCoffeMug"/>
214     </StackLayout.GestureRecognizers>
215 </StackLayout>
216 <StackLayout Grid.Column="0"
217     Grid.Row="5"
218     VerticalOptions="End"
219     Spacing="5"
220     Margin="40,0,40,15">
221     <StackLayout.GestureRecognizers>
222         <TapGestureRecognizer Tapped="NavigateToReCoffeMug"/>
223     </StackLayout.GestureRecognizers>
224     <Label Text="Use A Reusable Cup"
225         TextColor="White"
226         HorizontalOptions="Center"
227         FontSize="24"
228         FontAttributes="Bold"
229         FontFamily="Proxima Nova"/>
230     <Label Text="Use a reusable coffee mug and reduce littering from coffee
cups."
231         TextColor="White"
232         HorizontalOptions="Center"
233         FontSize="15"
234         FontFamily="Proxima Nova Thin"

```

```

235         HorizontalTextAlignment="Center"/>
236     </StackLayout>
237
238     <Image Grid.Column="0"
239           Grid.Row="6"
240           Aspect="AspectFill"
241           Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.FoodDelivered.jpg}"/>
242     <StackLayout Grid.Column="0"
243               Grid.Row="6"
244               BackgroundColor="Black"
245               Opacity=".5">
246         <StackLayout.GestureRecognizers>
247           <TapGestureRecognizer Tapped="NavigateToFoodDelivered"/>
248         </StackLayout.GestureRecognizers>
249     </StackLayout>
250     <StackLayout Grid.Column="0"
251               Grid.Row="6"
252               VerticalOptions="End"
253               Spacing="5"
254               Margin="40,0,40,15">
255         <StackLayout.GestureRecognizers>
256           <TapGestureRecognizer Tapped="NavigateToFoodDelivered"/>
257         </StackLayout.GestureRecognizers>
258         <Label Text="Have All Food Delivered"
259               TextColor="White"
260               HorizontalOptions="Center"
261               FontSize="24"
262               FontAttributes="Bold"
263               FontFamily="Proxima Nova"/>
264         <Label Text="Instead of traveling to the shops have all your food
delivered."
265               TextColor="White"
266               HorizontalOptions="Center"
267               FontSize="15"
268               FontFamily="Proxima Nova Thin"
269               HorizontalTextAlignment="Center"/>
270     </StackLayout>
271
272     <Image Grid.Column="0"
273           Grid.Row="7"
274           Aspect="AspectFill"
275           Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.OwnCoffee.jpg}"/>
276     <StackLayout Grid.Column="0"
277               Grid.Row="7"
278               BackgroundColor="Black"
279               Opacity=".5">
280         <StackLayout.GestureRecognizers>
281           <TapGestureRecognizer Tapped="NavigateToOwnCoffee"/>
282         </StackLayout.GestureRecognizers>
283     </StackLayout>
284     <StackLayout Grid.Column="0"
285               Grid.Row="7"
286               VerticalOptions="End"
287               Spacing="5"
288               Margin="40,0,40,15">
289         <StackLayout.GestureRecognizers>
290           <TapGestureRecognizer Tapped="NavigateToOwnCoffee"/>
291         </StackLayout.GestureRecognizers>
292         <Label Text="Brew Your Coffee"
293               TextColor="White"

```

```

294         HorizontalOptions="Center"
295         FontSize="24"
296         FontAttributes="Bold"
297         FontFamily="Proxima Nova"/>
298     <Label Text="Brew your own coffee instead of buying it."
299         TextColor="White"
300         HorizontalOptions="Center"
301         FontSize="15"
302         FontFamily="Proxima Nova Thin"
303         HorizontalTextAlignment="Center"/>
304 </StackLayout>
305
306 <Image Grid.Column="0"
307         Grid.Row="8"
308         Aspect="AspectFill"
309         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.SteeStraw.jpg}"/>
310 <StackLayout Grid.Column="0"
311         Grid.Row="8"
312         BackgroundColor="Black"
313         Opacity=".5">
314     <StackLayout.GestureRecognizers>
315         <TapGestureRecognizer Tapped="NavigateToSteelStraw"/>
316     </StackLayout.GestureRecognizers>
317 </StackLayout>
318 <StackLayout Grid.Column="0"
319         Grid.Row="8"
320         VerticalOptions="End"
321         Spacing="5"
322         Margin="40,0,40,15">
323     <StackLayout.GestureRecognizers>
324         <TapGestureRecognizer Tapped="NavigateToSteelStraw"/>
325     </StackLayout.GestureRecognizers>
326 <Label Text="Use Steel Straws"
327         TextColor="White"
328         HorizontalOptions="Center"
329         FontSize="24"
330         FontAttributes="Bold"
331         FontFamily="Proxima Nova"/>
332 <Label Text="Buy and use steel straws over plastic or paper ones to
reduce waste."
333         TextColor="White"
334         HorizontalOptions="Center"
335         FontSize="15"
336         FontFamily="Proxima Nova Thin"
337         HorizontalTextAlignment="Center"/>
338 </StackLayout>
339
340 <Image Grid.Column="0"
341         Grid.Row="9"
342         Aspect="AspectFill"
343         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"
344         Margin="0,0,0,5"/>
345 <StackLayout Grid.Column="0"
346         Grid.Row="9"
347         BackgroundColor="Black"
348         Opacity=".5"
349         Margin="0,0,0,5">
350     <StackLayout.GestureRecognizers>
351         <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
352 </StackLayout.GestureRecognizers>

```

```
353         </StackLayout>
354         <StackLayout Grid.Column="0"
355                     Grid.Row="9"
356                     VerticalOptions="End"
357                     Spacing="5"
358                     Margin="40,0,40,15">
359             <StackLayout.GestureRecognizers>
360                 <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
361             </StackLayout.GestureRecognizers>
362             <Label Text="Use A Reusable Bottle"
363                 TextColor="White"
364                 HorizontalOptions="Center"
365                 FontSize="24"
366                 FontAttributes="Bold"
367                 FontFamily="Proxima Nova"/>
368             <Label Text="Purchase and use a reusable water bottle instead of using
369 plastic bottles."
370                 TextColor="White"
371                 HorizontalOptions="Center"
372                 FontSize="15"
373                 FontFamily="Proxima Nova Thin"
374                 HorizontalTextAlignment="Center"/>
375         </StackLayout>
376     </Grid>
377 </ScrollView>
378 </ContentPage.Content>
</ContentPage>
```



```
1 /!* \class The FoodAndDrinkPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the FoodAndDrinkPage View Class. This page displays and allows the
   navigation to each of the actions in the FoodAndDrink category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.FoodAndDrink;
11 using Application_Green_Quake.Views.EcoActions.Water;
12 using System;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class FoodAndDrinkPage : ContentPage
20     {
21         public FoodAndDrinkPage()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function navigates to BuyOrganicFood.
27         */
28         private async void NavigateToBuyOrganicFood(object sender, EventArgs e)
29         {
30             await Navigation.PushAsync(new BuyOrganicFood());
31         }
32         /** This function navigates to WaterOverFizzy.
33         */
34         private async void NavigateToWaterOverFizzy(object sender, EventArgs e)
35         {
36             await Navigation.PushAsync(new WaterOverFizzy());
37         }
38         /** This function navigates to EatAllYouMake.
39         */
40         private async void NavigateToEatAllYouMake(object sender, EventArgs e)
41         {
42             await Navigation.PushAsync(new EatAllYouMake());
43         }
44         /** This function navigates to SaveLeftOvers.
45         */
46         private async void NavigateToSaveLeftovers(object sender, EventArgs e)
47         {
48             await Navigation.PushAsync(new SaveLeftOvers());
49         }
50         /** This function navigates to NoMeat.
51         */
52         private async void NavigateToNoMeat(object sender, EventArgs e)
53         {
54             await Navigation.PushAsync(new NoMeat());
55         }
56         /** This function navigates to ReCoffeeMug.
57         */
58         private async void NavigateToReCoffeMug(object sender, EventArgs e)
59         {
```



```
60     await Navigation.PushAsync(new ReCoffeeMug());
61 }
62 /** This function navigates to FoodDelivered.
63 */
64 private async void NavigateToFoodDelivered(object sender, EventArgs e)
65 {
66     await Navigation.PushAsync(new FoodDelivered());
67 }
68 /** This function navigates to OwnCoffee.
69 */
70 private async void NavigateToOwnCoffee(object sender, EventArgs e)
71 {
72     await Navigation.PushAsync(new OwnCoffee());
73 }
74 /** This function navigates to SteelStraw.
75 */
76 private async void NavigateToSteelStraw(object sender, EventArgs e)
77 {
78     await Navigation.PushAsync(new SteelStraw());
79 }
80 /** This function navigates to ReusableWater.
81 */
82 private async void NavigateToReusableWater(object sender, EventArgs e)
83 {
84     await Navigation.PushAsync(new ReusableWater());
85 }
86 /** 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
87     * and display it in the navigation bar.
88     */
89     protected override void OnAppearing()
90     {
91         GetData data = new GetData();
92         data.SetLvl();
93
94         theLevel.Text = "LVL: " + GetData.lvl.ToString();
95     }
96 }
97 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.HabitsPage"
6     xmlns:local="clr-namespace:Application_Green_Quake.Models"
7     Title="Habits Subcategories">
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Habits" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15            Margin="0,0,30,0"/>
16        </StackLayout>
17    </NavigationPage.TitleView>
18
19    <ContentPage.Content>
20        <ScrollView>
21            <Grid Margin="5,5,5,5">
22                <Grid.RowDefinitions>
23                    <RowDefinition Height="200" />
24                    <RowDefinition Height="200" />
25                    <RowDefinition Height="200" />
26                    <RowDefinition Height="200" />
27                    <RowDefinition Height="200" />
28                    <RowDefinition Height="200" />
29                </Grid.RowDefinitions>
30                <Grid.ColumnDefinitions>
31                    <ColumnDefinition />
32                </Grid.ColumnDefinitions>
33
34                <Image Grid.Column="0"
35                    Grid.Row="0"
36                    Aspect="AspectFill"
37                    Source="{local:ImageResource
38        Application_Green_Quake.Images.SubCategories.Habits.TimedBrushing.jpg}"/>
39                <StackLayout Grid.Column="0"
40                    Grid.Row="0"
41                    BackgroundColor="Black"
42                    Opacity=".5">
43                    <StackLayout.GestureRecognizers>
44                        <TapGestureRecognizer Tapped="NavigateToBrushingPage"/>
45                    </StackLayout.GestureRecognizers>
46                </StackLayout>
47                <StackLayout Grid.Column="0"
48                    Grid.Row="0"
49                    VerticalOptions="End"
50                    Spacing="5"
51                    Margin="40,0,40,15">
52                    <StackLayout.GestureRecognizers>
53                        <TapGestureRecognizer Tapped="NavigateToBrushingPage"/>
54                    </StackLayout.GestureRecognizers>
55                    <Label Text="Tap Off"
56                        TextColor="White"
57                        HorizontalOptions="Center"
58                        FontSize="24"
59                        FontAttributes="Bold"
60                        FontFamily="Proxima Nova"/>
61                    <Label Text="Turn off the tap when you are brushing your teeth and save

```

```

water and the planet."
57         TextColor="White"
58         HorizontalOptions="Center"
59         FontSize="15"
60         FontFamily="Proxima Nova Thin"
61         HorizontalTextAlignment="Center"/>
62     </StackLayout>
63
64     <Image Grid.Column="0"
65           Grid.Row="1"
66           Aspect="AspectFill"
67           Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.TimedShower.jpg}"/>
68     <StackLayout Grid.Column="0"
69           Grid.Row="1"
70           BackgroundColor="Black"
71           Opacity=".5">
72         <StackLayout.GestureRecognizers>
73             <TapGestureRecognizer Tapped="NavigateToTimedShowerPage"/>
74         </StackLayout.GestureRecognizers>
75     </StackLayout>
76     <StackLayout Grid.Column="0"
77           Grid.Row="1"
78           VerticalOptions="End"
79           Spacing="5"
80           Margin="40,0,40,15">
81         <StackLayout.GestureRecognizers>
82             <TapGestureRecognizer Tapped="NavigateToTimedShowerPage"/>
83         </StackLayout.GestureRecognizers>
84         <Label Text="Time Your Shower"
85               TextColor="White"
86               HorizontalOptions="Center"
87               FontSize="24"
88               FontAttributes="Bold"
89               FontFamily="Proxima Nova"/>
90         <Label Text="Time your showers so they don't take too long and save
water."
91               TextColor="White"
92               HorizontalOptions="Center"
93               FontSize="15"
94               FontFamily="Proxima Nova Thin"
95               HorizontalTextAlignment="Center"/>
96     </StackLayout>
97
98     <Image Grid.Column="0"
99           Grid.Row="2"
100          Aspect="AspectFill"
101          Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.ShowerNoBath.jpg}"/>
102     <StackLayout Grid.Column="0"
103           Grid.Row="2"
104           BackgroundColor="Black"
105           Opacity=".5">
106         <StackLayout.GestureRecognizers>
107             <TapGestureRecognizer Tapped="NavigateToShoweredInsteadOfBath"/>
108         </StackLayout.GestureRecognizers>
109     </StackLayout>
110     <StackLayout Grid.Column="0"
111           Grid.Row="2"
112           VerticalOptions="End"
113           Spacing="5"
114           Margin="40,0,40,15">

```

```

115         <StackLayout.GestureRecognizers>
116             <TapGestureRecognizer Tapped="NavigateToShoweredInsteadOfBath"/>
117         </StackLayout.GestureRecognizers>
118         <Label Text="Shower No Bath"
119             TextColor="White"
120             HorizontalOptions="Center"
121             FontSize="24"
122             FontAttributes="Bold"
123             FontFamily="Proxima Nova"/>
124         <Label Text="Take a brief shower over taking a bath."
125             TextColor="White"
126             HorizontalOptions="Center"
127             FontSize="15"
128             FontFamily="Proxima Nova Thin"
129             HorizontalTextAlignment="Center"/>
130     </StackLayout>
131
132     <Image Grid.Column="0"
133         Grid.Row="3"
134         Aspect="AspectFill"
135         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.FullDishwasher.jpg}"/>
136     <StackLayout Grid.Column="0"
137         Grid.Row="3"
138         BackgroundColor="Black"
139         Opacity=".5">
140         <StackLayout.GestureRecognizers>
141             <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
142         </StackLayout.GestureRecognizers>
143     </StackLayout>
144     <StackLayout Grid.Column="0"
145         Grid.Row="3"
146         VerticalOptions="End"
147         Spacing="5"
148         Margin="40,0,40,15">
149         <StackLayout.GestureRecognizers>
150             <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
151         </StackLayout.GestureRecognizers>
152         <Label Text="Full Dishwasher"
153             TextColor="White"
154             HorizontalOptions="Center"
155             FontSize="24"
156             FontAttributes="Bold"
157             FontFamily="Proxima Nova"/>
158         <Label Text="Only use the dishwasher when it is full."
159             TextColor="White"
160             HorizontalOptions="Center"
161             FontSize="15"
162             FontFamily="Proxima Nova Thin"
163             HorizontalTextAlignment="Center"/>
164     </StackLayout>
165
166     <Image Grid.Column="0"
167         Grid.Row="4"
168         Aspect="AspectFill"
169         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.OffLights.jpg}"/>
170     <StackLayout Grid.Column="0"
171         Grid.Row="4"
172         BackgroundColor="Black"
173         Opacity=".5">
174         <StackLayout.GestureRecognizers>

```

```

175         <TapGestureRecognizer Tapped="NavigateToTurnOffLights"/>
176     </StackLayout.GestureRecognizers>
177 </StackLayout>
178 <StackLayout Grid.Column="0"
179     Grid.Row="4"
180     VerticalOptions="End"
181     Spacing="5"
182     Margin="40,0,40,15">
183     <StackLayout.GestureRecognizers>
184         <TapGestureRecognizer Tapped="NavigateToTurnOffLights"/>
185     </StackLayout.GestureRecognizers>
186     <Label Text="Turn Off The Lights"
187         TextColor="White"
188         HorizontalOptions="Center"
189         FontSize="24"
190         FontAttributes="Bold"
191         FontFamily="Proxima Nova"/>
192     <Label Text="Turn off the lights when leaving the room and save
energy."
193         TextColor="White"
194         HorizontalOptions="Center"
195         FontSize="15"
196         FontFamily="Proxima Nova Thin"
197         HorizontalTextAlignment="Center"/>
198 </StackLayout>
199
200 <Image Grid.Column="0"
201     Grid.Row="5"
202     Aspect="AspectFill"
203     Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.MatchOverLighter.jpg}"
204     Margin="0,0,0,5"/>
205 <StackLayout Grid.Column="0"
206     Grid.Row="5"
207     BackgroundColor="Black"
208     Opacity=".5"
209     Margin="0,0,0,5">
210     <StackLayout.GestureRecognizers>
211         <TapGestureRecognizer Tapped="NavigateToUseMatches"/>
212     </StackLayout.GestureRecognizers>
213 </StackLayout>
214 <StackLayout Grid.Column="0"
215     Grid.Row="5"
216     VerticalOptions="End"
217     Spacing="5"
218     Margin="40,0,40,15">
219     <StackLayout.GestureRecognizers>
220         <TapGestureRecognizer Tapped="NavigateToUseMatches"/>
221     </StackLayout.GestureRecognizers>
222     <Label Text="Matches Over Lighters"
223         TextColor="White"
224         HorizontalOptions="Center"
225         FontSize="24"
226         FontAttributes="Bold"
227         FontFamily="Proxima Nova"/>
228     <Label Text="Use Matches over lighters as they are more environmentally
friendly."
229         TextColor="White"
230         HorizontalOptions="Center"
231         FontSize="15"
232         FontFamily="Proxima Nova Thin"
233         HorizontalTextAlignment="Center"/>

```

```
234         </StackLayout>
235     </Grid>
236 </ScrollView>
237 </ContentPage.Content>
238 </ContentPage>
```

```
1 /!* \class The HabitsPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the HabitsPage View Class. This page displays and allows the
   navigation to each of the actions in the Habits category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Habits;
11 using System;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class HabitsPage : ContentPage
19     {
20         public HabitsPage()
21         {
22             InitializeComponent();
23             OnAppearing();
24         }
25         /** This function navigates to BrushingTeeth.
26         */
27         private async void NavigateToBrushingPage(object sender, EventArgs e)
28         {
29             await Navigation.PushAsync(new BrushingTeeth());
30         }
31         /** This function navigates to TimedShower.
32         */
33         private async void NavigateToTimedShowerPage(object sender, EventArgs e)
34         {
35             await Navigation.PushAsync(new TimedShower());
36         }
37         /** This function navigates to ShowerInstead.
38         */
39         private async void NavigateToShoweredInsteadOfBath(object sender, EventArgs e)
40         {
41             await Navigation.PushAsync(new ShowerInstead());
42         }
43         /** This function navigates to DishwasherFull.
44         */
45         private async void NavigateToDishwasherFull(object sender, EventArgs e)
46         {
47             await Navigation.PushAsync(new DishwasherFull());
48         }
49         }
50         /** This function navigates to TurnOffLights.
51         */
52         private async void NavigateToTurnOffLights(object sender, EventArgs e)
53         {
54             await Navigation.PushAsync(new TurnOffLights());
55         }
56         }
57         /** This function navigates to UseMatches.
58         */
59         private async void NavigateToUseMatches(object sender, EventArgs e)
```

```
60     {
61         await Navigation.PushAsync(new UseMatches());
62     }
63     /** 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
64     * and display it in the navigation bar.
65     */
66     protected override void OnAppearing()
67     {
68         GetData data = new GetData();
69         data.SetLvl();
70
71         theLevel.Text = "LVL: " + GetData.lvl.ToString();
72     }
73 }
74 }
```



```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.HomePage"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models"
6     Title="Home Subcategories">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Home" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15            Margin="0,0,30,0"/>
16         </StackLayout>
17     </NavigationPage.TitleView>
18
19     <ContentPage.Content>
20         <ScrollView>
21             <Grid Margin="5,5,5,5">
22                 <Grid.RowDefinitions>
23                     <RowDefinition Height="200" />
24                     <RowDefinition Height="200" />
25                     <RowDefinition Height="200" />
26                     <RowDefinition Height="200" />
27                     <RowDefinition Height="200" />
28                     <RowDefinition Height="200" />
29                     <RowDefinition Height="200" />
30                 </Grid.RowDefinitions>
31                 <Grid.ColumnDefinitions>
32                     <ColumnDefinition />
33                 </Grid.ColumnDefinitions>
34
35                 <Image Grid.Column="0"
36                    Grid.Row="0"
37                    Aspect="AspectFill"
38                    Source="{local:ImageResource
39        Application_Green_Quake.Images.SubCategories.Home.AirOut.jpg}"/>
40                 <StackLayout Grid.Column="0"
41                    Grid.Row="0"
42                    BackgroundColor="Black"
43                    Opacity=".5">
44                     <StackLayout.GestureRecognizers>
45                         <TapGestureRecognizer Tapped="NavigateToAirOutHome"/>
46                     </StackLayout.GestureRecognizers>
47                 </StackLayout>
48                 <StackLayout Grid.Column="0"
49                    Grid.Row="0"
50                    VerticalOptions="End"
51                    Spacing="5"
52                    Margin="40,0,40,15">
53                     <StackLayout.GestureRecognizers>
54                         <TapGestureRecognizer Tapped="NavigateToAirOutHome"/>
55                     </StackLayout.GestureRecognizers>
56                     <Label Text="Air Out Your Home"
57                        TextColor="White"
58                        HorizontalOptions="Center"
59                        FontSize="24"
60                        FontAttributes="Bold"
61                        FontFamily="Proxima Nova"/>
62                     <Label Text="Air out your home. Let some fresh air inside."

```

```
58         TextColor="White"
59         HorizontalOptions="Center"
60         FontSize="15"
61         FontFamily="Proxima Nova Thin"
62         HorizontalTextAlignment="Center"/>
63     </StackLayout>
64
65     <Image Grid.Column="0"
66         Grid.Row="1"
67         Aspect="AspectFill"
68         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Home.Outside.jpg}"/>
69     <StackLayout Grid.Column="0"
70         Grid.Row="1"
71         BackgroundColor="Black"
72         Opacity=".5">
73         <StackLayout.GestureRecognizers>
74             <TapGestureRecognizer Tapped="NavigateToOutsideOnce"/>
75         </StackLayout.GestureRecognizers>
76     </StackLayout>
77     <StackLayout Grid.Column="0"
78         Grid.Row="1"
79         VerticalOptions="End"
80         Spacing="5"
81         Margin="40,0,40,15">
82         <StackLayout.GestureRecognizers>
83             <TapGestureRecognizer Tapped="NavigateToOutsideOnce"/>
84         </StackLayout.GestureRecognizers>
85         <Label Text="Go Outside"
86             TextColor="White"
87             HorizontalOptions="Center"
88             FontSize="24"
89             FontAttributes="Bold"
90             FontFamily="Proxima Nova"/>
91         <Label Text="When brushing your teeth it can be easy to leave the water
running but this attributes to major water waste."
92             TextColor="White"
93             HorizontalOptions="Center"
94             FontSize="15"
95             FontFamily="Proxima Nova Thin"
96             HorizontalTextAlignment="Center"/>
97     </StackLayout>
98
99     <Image Grid.Column="0"
100         Grid.Row="2"
101         Aspect="AspectFill"
102         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Home.PlantHome.jpg}"/>
103     <StackLayout Grid.Column="0"
104         Grid.Row="2"
105         BackgroundColor="Black"
106         Opacity=".5">
107         <StackLayout.GestureRecognizers>
108             <TapGestureRecognizer Tapped="NavigateToPlantIntoHome"/>
109         </StackLayout.GestureRecognizers>
110     </StackLayout>
111     <StackLayout Grid.Column="0"
112         Grid.Row="2"
113         VerticalOptions="End"
114         Spacing="5"
115         Margin="40,0,40,15">
116         <StackLayout.GestureRecognizers>
```

```

117         <TapGestureRecognizer Tapped="NavigateToPlantIntoHome"/>
118     </StackLayout.GestureRecognizers>
119     <Label Text="Bring A Plant Inside"
120         TextColor="White"
121         HorizontalOptions="Center"
122         FontSize="24"
123         FontAttributes="Bold"
124         FontFamily="Proxima Nova"/>
125     <Label Text="Bring a plant into your home."
126         TextColor="White"
127         HorizontalOptions="Center"
128         FontSize="15"
129         FontFamily="Proxima Nova Thin"
130         HorizontalTextAlignment="Center"/>
131 </StackLayout>
132
133     <Image Grid.Column="0"
134         Grid.Row="3"
135         Aspect="AspectFill"
136         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Home.NonHarmful.jpg}"/>
137     <StackLayout Grid.Column="0"
138         Grid.Row="3"
139         BackgroundColor="Black"
140         Opacity=".5">
141         <StackLayout.GestureRecognizers>
142             <TapGestureRecognizer Tapped="NavigateToNonHarmfulProducts"/>
143         </StackLayout.GestureRecognizers>
144     </StackLayout>
145     <StackLayout Grid.Column="0"
146         Grid.Row="3"
147         VerticalOptions="End"
148         Spacing="5"
149         Margin="40,0,40,15">
150         <StackLayout.GestureRecognizers>
151             <TapGestureRecognizer Tapped="NavigateToNonHarmfulProducts"/>
152         </StackLayout.GestureRecognizers>
153         <Label Text="Use Non Harmful Products"
154             TextColor="White"
155             HorizontalOptions="Center"
156             FontSize="24"
157             FontAttributes="Bold"
158             FontFamily="Proxima Nova"
159             HorizontalTextAlignment="Center"/>
160         <Label Text="Use non harmful bio products instead of harmful ones when
possible."
161             TextColor="White"
162             HorizontalOptions="Center"
163             FontSize="15"
164             FontFamily="Proxima Nova Thin"
165             HorizontalTextAlignment="Center"/>
166     </StackLayout>
167
168     <Image Grid.Column="0"
169         Grid.Row="4"
170         Aspect="AspectFill"
171         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Home.Flush.jpg}"/>
172     <StackLayout Grid.Column="0"
173         Grid.Row="4"
174         BackgroundColor="Black"
175         Opacity=".5">

```

```

176         <StackLayout.GestureRecognizers>
177             <TapGestureRecognizer Tapped="NavigateToSaveFlush"/>
178         </StackLayout.GestureRecognizers>
179     </StackLayout>
180     <StackLayout Grid.Column="0"
181         Grid.Row="4"
182         VerticalOptions="End"
183         Spacing="5"
184         Margin="40,0,40,15">
185         <StackLayout.GestureRecognizers>
186             <TapGestureRecognizer Tapped="NavigateToSaveFlush"/>
187         </StackLayout.GestureRecognizers>
188         <Label Text="Save A Flush"
189             TextColor="White"
190             HorizontalOptions="Center"
191             FontSize="24"
192             FontAttributes="Bold"
193             FontFamily="Proxima Nova"/>
194         <Label Text="Save a flush when you can and save water."
195             TextColor="White"
196             HorizontalOptions="Center"
197             FontSize="15"
198             FontFamily="Proxima Nova Thin"
199             HorizontalTextAlignment="Center"/>
200     </StackLayout>
201
202     <Image Grid.Column="0"
203         Grid.Row="5"
204         Aspect="AspectFill"
205         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.Napkin.jpg}"/>
206     <StackLayout Grid.Column="0"
207         Grid.Row="5"
208         BackgroundColor="Black"
209         Opacity=".5">
210         <StackLayout.GestureRecognizers>
211             <TapGestureRecognizer Tapped="NavigateToClothNapkins"/>
212         </StackLayout.GestureRecognizers>
213     </StackLayout>
214     <StackLayout Grid.Column="0"
215         Grid.Row="5"
216         VerticalOptions="End"
217         Spacing="5"
218         Margin="40,0,40,15">
219         <StackLayout.GestureRecognizers>
220             <TapGestureRecognizer Tapped="NavigateToClothNapkins"/>
221         </StackLayout.GestureRecognizers>
222         <Label Text="Cloth Napkins"
223             TextColor="White"
224             HorizontalOptions="Center"
225             FontSize="24"
226             FontAttributes="Bold"
227             FontFamily="Proxima Nova"/>
228         <Label Text="Use Cloth Napkins over paper ones."
229             TextColor="White"
230             HorizontalOptions="Center"
231             FontSize="15"
232             FontFamily="Proxima Nova Thin"
233             HorizontalTextAlignment="Center"/>
234     </StackLayout>
235

```

```
236         <Image Grid.Column="0"
237             Grid.Row="6"
238             Aspect="AspectFill"
239             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.Towel.jpg}"
240             Margin="0,0,0,5"/>
241         <StackLayout Grid.Column="0"
242             Grid.Row="6"
243             BackgroundColor="Black"
244             Opacity=".5"
245             Margin="0,0,0,5">
246             <StackLayout.GestureRecognizers
247                 <TapGestureRecognizer Tapped="NavigateToClothTowels"/>
248             </StackLayout.GestureRecognizers>
249         </StackLayout>
250         <StackLayout Grid.Column="0"
251             Grid.Row="6"
252             VerticalOptions="End"
253             Spacing="5"
254             Margin="40,0,40,15">
255             <StackLayout.GestureRecognizers>
256                 <TapGestureRecognizer Tapped="NavigateToClothTowels"/>
257             </StackLayout.GestureRecognizers>
258             <Label Text="Cloth Towels"
259                 TextColor="White"
260                 HorizontalOptions="Center"
261                 FontSize="24"
262                 FontAttributes="Bold"
263                 FontFamily="Proxima Nova"/>
264             <Label Text="Use Cloth Towels over paper ones."
265                 TextColor="White"
266                 HorizontalOptions="Center"
267                 FontSize="15"
268                 FontFamily="Proxima Nova Thin"
269                 HorizontalTextAlignment="Center"/>
270         </StackLayout>
271     </Grid>
272 </ScrollView>
273 </ContentPage.Content>
274 </ContentPage>
```

```
1 /!* \class The HomePage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the HomePage View Class. This page displays and allows the
   navigation to each of the actions in the Home category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Home;
11 using Application_Green_Quake.Views.EcoActions.Shopping;
12 using System;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class HomePage : ContentPage
20     {
21         public HomePage()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function navigates to AirOutHome.
27         */
28         private async void NavigateToAirOutHome(object sender, EventArgs e)
29         {
30             await Navigation.PushAsync(new AirOutHome());
31         }
32         /** This function navigates to OutsideOnce.
33         */
34         private async void NavigateToOutsideOnce(object sender, EventArgs e)
35         {
36             await Navigation.PushAsync(new OutsideOnce());
37         }
38         /** This function navigates to PlantIntoHome.
39         */
40         private async void NavigateToPlantIntoHome(object sender, EventArgs e)
41         {
42             await Navigation.PushAsync(new PlantIntoHome());
43         }
44         /** This function navigates to NonHarmfulProducts.
45         */
46         private async void NavigateToNonHarmfulProducts(object sender, EventArgs e)
47         {
48             await Navigation.PushAsync(new NonHarmfulProducts());
49         }
50         /** This function navigates to ToiletFlushes.
51         */
52         private async void NavigateToSaveFlush(object sender, EventArgs e)
53         {
54             await Navigation.PushAsync(new ToiletFlushes());
55         }
56     }
57 }
58
59
```

```
60     }
61     /** This function navigates to ClothNapkins.
62     */
63     private async void NavigateToClothNapkins(object sender, EventArgs e)
64     {
65         await Navigation.PushAsync(new ClothNapkins());
66     }
67     /** This function navigates to ClothTowels.
68     */
69     private async void NavigateToClothTowels(object sender, EventArgs e)
70     {
71         await Navigation.PushAsync(new ClothTowels());
72     }
73     /** This function is called before the page is displayed and it created an object
74     ans uses it's SetLvl method to set the players level in the app
75     * and display it in the navigation bar.
76     */
77     protected override void OnAppearing()
78     {
79         GetData data = new GetData();
80         data.SetLvl();
81
82         theLevel.Text = "LVL: " + GetData.lvl.ToString();
83     }
84 }
85 }
86 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.OutdoorsPage"
6     xmlns:local="clr-namespace:Application_Green_Quake.Models"
7     Title="Outdoors Subcategories">
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Outdoors" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15            Margin="0,0,30,0"/>
16        </StackLayout>
17    </NavigationPage.TitleView>
18
19    <ContentPage.Content>
20        <ScrollView>
21            <Grid Margin="5,5,5,5">
22                <Grid.RowDefinitions>
23                    <RowDefinition Height="200" />
24                    <RowDefinition Height="200" />
25                    <RowDefinition Height="200" />
26                    <RowDefinition Height="200" />
27                    <RowDefinition Height="200" />
28                    <RowDefinition Height="200" />
29                    <RowDefinition Height="200" />
30                    <RowDefinition Height="200" />
31                    <RowDefinition Height="200" />
32                    <RowDefinition Height="200" />
33                </Grid.RowDefinitions>
34                <Grid.ColumnDefinitions>
35                    <ColumnDefinition />
36                </Grid.ColumnDefinitions>
37
38                <Image Grid.Column="0"
39                Grid.Row="0"
40                Aspect="AspectFill"
41                Source="{local:ImageResource
42                Application_Green_Quake.Images.SubCategories.Outdoors.Tree.jpg}"/>
43                <StackLayout Grid.Column="0"
44                Grid.Row="0"
45                BackgroundColor="Black"
46                Opacity=".5">
47                    <StackLayout.GestureRecognizers>
48                        <TapGestureRecognizer Tapped="NavigateToPlantATree"/>
49                    </StackLayout.GestureRecognizers>
50                </StackLayout>
51                <StackLayout Grid.Column="0"
52                Grid.Row="0"
53                VerticalOptions="End"
54                Spacing="5"
55                Margin="40,0,40,15">
56                    <StackLayout.GestureRecognizers>
57                        <TapGestureRecognizer Tapped="NavigateToPlantATree"/>
58                    </StackLayout.GestureRecognizers>
59                    <Label Text="Plant A Tree"
60                    TextColor="White"

```



```

57         HorizontalOptions="Center"
58         FontSize="24"
59         FontAttributes="Bold"
60         FontFamily="Proxima Nova"/>
61     <Label Text="Plant a tree somewhere you can and watch it grow."
62         TextColor="White"
63         HorizontalOptions="Center"
64         FontSize="15"
65         FontFamily="Proxima Nova Thin"
66         HorizontalTextAlignment="Center"/>
67 </StackLayout>
68
69     <Image Grid.Column="0"
70         Grid.Row="1"
71         Aspect="AspectFill"
72         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.Flower.jpg}"/>
73     <StackLayout Grid.Column="0"
74         Grid.Row="1"
75         BackgroundColor="Black"
76         Opacity=".5">
77         <StackLayout.GestureRecognizers>
78             <TapGestureRecognizer Tapped="NavigateToPlantAFlower"/>
79         </StackLayout.GestureRecognizers>
80     </StackLayout>
81     <StackLayout Grid.Column="0"
82         Grid.Row="1"
83         VerticalOptions="End"
84         Spacing="5"
85         Margin="40,0,40,15">
86         <StackLayout.GestureRecognizers>
87             <TapGestureRecognizer Tapped="NavigateToPlantAFlower"/>
88         </StackLayout.GestureRecognizers>
89         <Label Text="Plant A Flower"
90             TextColor="White"
91             HorizontalOptions="Center"
92             FontSize="24"
93             FontAttributes="Bold"
94             FontFamily="Proxima Nova"/>
95         <Label Text="Plant a flower in your garden. It doesn't just look good
it improves the environment"
96             TextColor="White"
97             HorizontalOptions="Center"
98             FontSize="15"
99             FontFamily="Proxima Nova Thin"
100            HorizontalTextAlignment="Center"/>
101     </StackLayout>
102
103     <Image Grid.Column="0"
104         Grid.Row="2"
105         Aspect="AspectFill"
106         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.Bush.jpg}"/>
107     <StackLayout Grid.Column="0"
108         Grid.Row="2"
109         BackgroundColor="Black"
110         Opacity=".5">
111         <StackLayout.GestureRecognizers>
112             <TapGestureRecognizer Tapped="NavigateToPlantABush"/>
113         </StackLayout.GestureRecognizers>
114     </StackLayout>
115     <StackLayout Grid.Column="0"

```

```
116         Grid.Row="2"
117         VerticalOptions="End"
118         Spacing="5"
119         Margin="40,0,40,15">
120     <StackLayout.GestureRecognizers>
121         <TapGestureRecognizer Tapped="NavigateToPlantABush"/>
122     </StackLayout.GestureRecognizers>
123     <Label Text="Plant A Bush"
124         TextColor="White"
125         HorizontalOptions="Center"
126         FontSize="24"
127         FontAttributes="Bold"
128         FontFamily="Proxima Nova"/>
129     <Label Text="Plant a bush and let it grow."
130         TextColor="White"
131         HorizontalOptions="Center"
132         FontSize="15"
133         FontFamily="Proxima Nova Thin"
134         HorizontalTextAlignment="Center"/>
135 </StackLayout>
136
137     <Image Grid.Column="0"
138         Grid.Row="3"
139         Aspect="AspectFill"
140         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.Picnic.jpg}"/>
141     <StackLayout Grid.Column="0"
142         Grid.Row="3"
143         BackgroundColor="Black"
144         Opacity=".5">
145         <StackLayout.GestureRecognizers>
146             <TapGestureRecognizer Tapped="NavigateToPicnic"/>
147         </StackLayout.GestureRecognizers>
148     </StackLayout>
149     <StackLayout Grid.Column="0"
150         Grid.Row="3"
151         VerticalOptions="End"
152         Spacing="5"
153         Margin="40,0,40,15">
154     <StackLayout.GestureRecognizers>
155         <TapGestureRecognizer Tapped="NavigateToPicnic"/>
156     </StackLayout.GestureRecognizers>
157     <Label Text="Have A Picnic"
158         TextColor="White"
159         HorizontalOptions="Center"
160         FontSize="24"
161         FontAttributes="Bold"
162         FontFamily="Proxima Nova"/>
163     <Label Text="Go outside and Have a Picnic."
164         TextColor="White"
165         HorizontalOptions="Center"
166         FontSize="15"
167         FontFamily="Proxima Nova Thin"
168         HorizontalTextAlignment="Center"/>
169 </StackLayout>
170
171     <Image Grid.Column="0"
172         Grid.Row="4"
173         Aspect="AspectFill"
174         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.Camping.jpg}"/>
175     <StackLayout Grid.Column="0"
```

```

176         Grid.Row="4"
177         BackgroundColor="Black"
178         Opacity=".5">
179         <StackLayout.GestureRecognizers>
180             <TapGestureRecognizer Tapped="NavigateToGoCamping"/>
181         </StackLayout.GestureRecognizers>
182     </StackLayout>
183     <StackLayout Grid.Column="0"
184         Grid.Row="4"
185         VerticalOptions="End"
186         Spacing="5"
187         Margin="40,0,40,15">
188         <StackLayout.GestureRecognizers>
189             <TapGestureRecognizer Tapped="NavigateToGoCamping"/>
190         </StackLayout.GestureRecognizers>
191         <Label Text="Go Camping"
192             TextColor="White"
193             HorizontalOptions="Center"
194             FontSize="24"
195             FontAttributes="Bold"
196             FontFamily="Proxima Nova"/>
197         <Label Text="Go Camping and have some fun. Use reusable gear."
198             TextColor="White"
199             HorizontalOptions="Center"
200             FontSize="15"
201             FontFamily="Proxima Nova Thin"
202             HorizontalTextAlignment="Center"/>
203     </StackLayout>
204
205     <Image Grid.Column="0"
206         Grid.Row="5"
207         Aspect="AspectFill"
208         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.Scoop.jpg}"/>
209     <StackLayout Grid.Column="0"
210         Grid.Row="5"
211         BackgroundColor="Black"
212         Opacity=".5">
213         <StackLayout.GestureRecognizers>
214             <TapGestureRecognizer Tapped="NavigateToScoop"/>
215         </StackLayout.GestureRecognizers>
216     </StackLayout>
217     <StackLayout Grid.Column="0"
218         Grid.Row="5"
219         VerticalOptions="End"
220         Spacing="5"
221         Margin="40,0,40,15">
222         <StackLayout.GestureRecognizers>
223             <TapGestureRecognizer Tapped="NavigateToScoop"/>
224         </StackLayout.GestureRecognizers>
225         <Label Text="Scoop da Poop"
226             TextColor="White"
227             HorizontalOptions="Center"
228             FontSize="24"
229             FontAttributes="Bold"
230             FontFamily="Proxima Nova"/>
231         <Label Text="When your dog makes a mess clean it up. It's only right."
232             TextColor="White"
233             HorizontalOptions="Center"
234             FontSize="15"
235             FontFamily="Proxima Nova Thin"

```

```
236         HorizontalTextAlignment="Center"/>
237     </StackLayout>
238
239     <Image Grid.Column="0"
240         Grid.Row="6"
241         Aspect="AspectFill"
242         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.HerbGarden.jpg}"/>
243     <StackLayout Grid.Column="0"
244         Grid.Row="6"
245         BackgroundColor="Black"
246         Opacity=".5">
247         <StackLayout.GestureRecognizers>
248             <TapGestureRecognizer Tapped="NavigateToSetUpHerbGarden"/>
249         </StackLayout.GestureRecognizers>
250     </StackLayout>
251     <StackLayout Grid.Column="0"
252         Grid.Row="6"
253         VerticalOptions="End"
254         Spacing="5"
255         Margin="40,0,40,15">
256         <StackLayout.GestureRecognizers>
257             <TapGestureRecognizer Tapped="NavigateToSetUpHerbGarden"/>
258         </StackLayout.GestureRecognizers>
259         <Label Text="Set Up A Herb Garden"
260             TextColor="White"
261             HorizontalOptions="Center"
262             FontSize="24"
263             FontAttributes="Bold"
264             FontFamily="Proxima Nova"/>
265         <Label Text="Set up a herb garden and plant some herbs."
266             TextColor="White"
267             HorizontalOptions="Center"
268             FontSize="15"
269             FontFamily="Proxima Nova Thin"
270             HorizontalTextAlignment="Center"/>
271     </StackLayout>
272
273     <Image Grid.Column="0"
274         Grid.Row="7"
275         Aspect="AspectFill"
276         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.VegGarden.jpg}"/>
277     <StackLayout Grid.Column="0"
278         Grid.Row="7"
279         BackgroundColor="Black"
280         Opacity=".5">
281         <StackLayout.GestureRecognizers>
282             <TapGestureRecognizer Tapped="NavigateToSetUpVegetableGarden"/>
283         </StackLayout.GestureRecognizers>
284     </StackLayout>
285     <StackLayout Grid.Column="0"
286         Grid.Row="7"
287         VerticalOptions="End"
288         Spacing="5"
289         Margin="40,0,40,15">
290         <StackLayout.GestureRecognizers>
291             <TapGestureRecognizer Tapped="NavigateToSetUpVegetableGarden"/>
292         </StackLayout.GestureRecognizers>
293         <Label Text="Set Up A Vegetable Garden"
294             TextColor="White"
295             HorizontalOptions="Center"
```

```
296         FontSize="24"
297         FontAttributes="Bold"
298         FontFamily="Proxima Nova"
299         HorizontalTextAlignment="Center"/>
300     <Label Text="Set up a Vegetable garden and plant some vegetable."
301           TextColor="White"
302           HorizontalOptions="Center"
303           FontSize="15"
304           FontFamily="Proxima Nova Thin"
305           HorizontalTextAlignment="Center"/>
306 </StackLayout>
307
308 <Image Grid.Column="0"
309        Grid.Row="8"
310        Aspect="AspectFill"
311        Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.FruitGarden.jpg}"/>
312 <StackLayout Grid.Column="0"
313             Grid.Row="8"
314             BackgroundColor="Black"
315             Opacity=".5">
316     <StackLayout.GestureRecognizers>
317         <TapGestureRecognizer Tapped="NavigateToSetUpFruitGarden"/>
318     </StackLayout.GestureRecognizers>
319 </StackLayout>
320 <StackLayout Grid.Column="0"
321             Grid.Row="8"
322             VerticalOptions="End"
323             Spacing="5"
324             Margin="40,0,40,15">
325     <StackLayout.GestureRecognizers>
326         <TapGestureRecognizer Tapped="NavigateToSetUpFruitGarden"/>
327     </StackLayout.GestureRecognizers>
328     <Label Text="Set Up A Fruit Garden"
329           TextColor="White"
330           HorizontalOptions="Center"
331           FontSize="24"
332           FontAttributes="Bold"
333           FontFamily="Proxima Nova"/>
334     <Label Text="Set up a Fruit garden and plant some fruit."
335           TextColor="White"
336           HorizontalOptions="Center"
337           FontSize="15"
338           FontFamily="Proxima Nova Thin"
339           HorizontalTextAlignment="Center"/>
340 </StackLayout>
341
342 <Image Grid.Column="0"
343        Grid.Row="9"
344        Aspect="AspectFill"
345        Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Water.RainBarrel.jpg}"/>
346 <StackLayout Grid.Column="0"
347             Grid.Row="9"
348             BackgroundColor="Black"
349             Opacity=".5">
350     <StackLayout.GestureRecognizers>
351         <TapGestureRecognizer Tapped="NavigateToSetUpRainWaterColector"/>
352     </StackLayout.GestureRecognizers>
353 </StackLayout>
354 <StackLayout Grid.Column="0"
```

```

355         Grid.Row="9"
356         VerticalOptions="End"
357         Spacing="5"
358         Margin="40,0,40,15">
359     <StackLayout.GestureRecognizers>
360         <TapGestureRecognizer Tapped="NavigateToSetUpRainWaterColector"/>
361     </StackLayout.GestureRecognizers>
362     <Label Text="Set Up A Rain Barrel"
363         TextColor="White"
364         HorizontalOptions="Center"
365         FontSize="24"
366         FontAttributes="Bold"
367         FontFamily="Proxima Nova"/>
368     <Label Text="Set up a rain barrel and collect rain water that you can
use."
369         TextColor="White"
370         HorizontalOptions="Center"
371         FontSize="15"
372         FontFamily="Proxima Nova Thin"
373         HorizontalTextAlignment="Center"/>
374 </StackLayout>
375
376     <Image Grid.Column="0"
377         Grid.Row="10"
378         Aspect="AspectFill"
379         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Outdoors.BirdFeeder.jpg}"
380         Margin="0,0,0,5"/>
381     <StackLayout Grid.Column="0"
382         Grid.Row="10"
383         BackgroundColor="Black"
384         Opacity=".5"
385         Margin="0,0,0,5">
386         <StackLayout.GestureRecognizers>
387             <TapGestureRecognizer Tapped="NavigateToSetUpBirdfeeder"/>
388         </StackLayout.GestureRecognizers>
389     </StackLayout>
390     <StackLayout Grid.Column="0"
391         Grid.Row="10"
392         VerticalOptions="End"
393         Spacing="5"
394         Margin="40,0,40,15">
395     <StackLayout.GestureRecognizers>
396         <TapGestureRecognizer Tapped="NavigateToSetUpBirdfeeder"/>
397     </StackLayout.GestureRecognizers>
398     <Label Text="Set Up A Bird Feeder"
399         TextColor="White"
400         HorizontalOptions="Center"
401         FontSize="24"
402         FontAttributes="Bold"
403         FontFamily="Proxima Nova"/>
404     <Label Text="Set up a bird feeder in your garden and help out the
wildlife."
405         TextColor="White"
406         HorizontalOptions="Center"
407         FontSize="15"
408         FontFamily="Proxima Nova Thin"
409         HorizontalTextAlignment="Center"/>
410 </StackLayout>
411 </Grid>
412 </ScrollView>
413 </ContentPage.Content>

```



```
1  /*! \class The OutdoorsPage View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the OutdoorsPage View Class. This page displays and allows the
   navigation to each of the actions in the Outdoors category.
7  *
8  */
9  using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Outdoors;
11 using Application_Green_Quake.Views.EcoActions.Water;
12 using System;
13 using Xamarin.Forms;
14 using Xamarin.Forms.Xaml;
15
16 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OutdoorsPage : ContentPage
20     {
21         public OutdoorsPage()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function navigates to PlantATree.
27         */
28         private async void NavigateToPlantATree(object sender, EventArgs e)
29         {
30             await Navigation.PushAsync(new PlantATree());
31         }
32         /** This function navigates to PlantAFlower.
33         */
34         private async void NavigateToPlantAFlower(object sender, EventArgs e)
35         {
36             await Navigation.PushAsync(new PlantAFlower());
37         }
38         /** This function navigates to PlantABush.
39         */
40         private async void NavigateToPlantABush(object sender, EventArgs e)
41         {
42             await Navigation.PushAsync(new PlantABush());
43         }
44         /** This function navigates to Picnic.
45         */
46         private async void NavigateToPicnic(object sender, EventArgs e)
47         {
48             await Navigation.PushAsync(new Picnic());
49         }
50         /** This function navigates to GoCamping.
51         */
52         private async void NavigateToGoCamping(object sender, EventArgs e)
53         {
54             await Navigation.PushAsync(new GoCamping());
55         }
56         /** This function navigates to Scoop.
57         */
58         private async void NavigateToScoop(object sender, EventArgs e)
59         {
```



```
60     await Navigation.PushAsync(new Scoop());
61 }
62 /** This function navigates to SetUpHerbGarden.
63 */
64 private async void NavigateToSetUpHerbGarden(object sender, EventArgs e)
65 {
66     await Navigation.PushAsync(new SetUpHerbGarden());
67 }
68 /** This function navigates to SetUpVegetableGarden.
69 */
70 private async void NavigateToSetUpVegetableGarden(object sender, EventArgs e)
71 {
72     await Navigation.PushAsync(new SetUpVegetableGarden());
73 }
74 /** This function navigates to SetUpFruitGarden.
75 */
76 private async void NavigateToSetUpFruitGarden(object sender, EventArgs e)
77 {
78     await Navigation.PushAsync(new SetUpFruitGarden());
79 }
80 /** This function navigates to RainBarrel.
81 */
82 private async void NavigateToSetUpRainWaterColector(object sender, EventArgs e)
83 {
84     await Navigation.PushAsync(new RainBarrel());
85 }
86 /** This function navigates to UpBirdfeeder.
87 */
88 private async void NavigateToSetUpBirdfeeder(object sender, EventArgs e)
89 {
90     await Navigation.PushAsync(new UpBirdfeeder());
91 }
92 /** This function is called before the page is displayed and it created an object
93 ans uses it's SetLvl method to set the players level in the app
94 * and display it in the navigation bar.
95 */
96 protected override void OnAppearing()
97 {
98     GetData data = new GetData();
99     data.SetLvl();
100
101     theLevel.Text = "LVL: " + GetData.lvl.ToString();
102 }
103 }
```



```

57         <Label Text="Use A Reusable Bottle"
58             TextColor="White"
59             HorizontalOptions="Center"
60             FontSize="24"
61             FontAttributes="Bold"
62             FontFamily="Proxima Nova"/>
63         <Label Text="Purchase and use a reusable water bottle instead of using
64 plastic bottles."
65             TextColor="White"
66             HorizontalOptions="Center"
67             FontSize="15"
68             FontFamily="Proxima Nova Thin"
69             HorizontalTextAlignment="Center"/>
70     </StackLayout>
71     <Image Grid.Column="0"
72         Grid.Row="1"
73         Aspect="AspectFill"
74         Source="{local:ImageResource
75 Application_Green_Quake.Images.SubCategories.Shopping.ReBag.jpg}"/>
76     <StackLayout Grid.Column="0"
77         Grid.Row="1"
78         BackgroundColor="Black"
79         Opacity=".5">
80         <StackLayout.GestureRecognizers>
81             <TapGestureRecognizer Tapped="NavigateToPurchaseReusableBag"/>
82         </StackLayout.GestureRecognizers>
83     </StackLayout>
84     <StackLayout Grid.Column="0"
85         Grid.Row="1"
86         VerticalOptions="End"
87         Spacing="5"
88         Margin="40,0,40,15">
89         <StackLayout.GestureRecognizers>
90             <TapGestureRecognizer Tapped="NavigateToPurchaseReusableBag"/>
91         </StackLayout.GestureRecognizers>
92         <Label Text="Reusable Bag"
93             TextColor="White"
94             HorizontalOptions="Center"
95             FontSize="24"
96             FontAttributes="Bold"
97             FontFamily="Proxima Nova"/>
98         <Label Text="Purchase and use a reusable bag. Don't use bags that you
99 throw away."
100             TextColor="White"
101             HorizontalOptions="Center"
102             FontSize="15"
103             FontFamily="Proxima Nova Thin"
104             HorizontalTextAlignment="Center"/>
105     </StackLayout>
106     <Image Grid.Column="0"
107         Grid.Row="2"
108         Aspect="AspectFill"
109         Source="{local:ImageResource
110 Application_Green_Quake.Images.SubCategories.Shopping.LocalProduct.jpg}"/>
111     <StackLayout Grid.Column="0"
112         Grid.Row="2"
113         BackgroundColor="Black"
114         Opacity=".5">
115         <StackLayout.GestureRecognizers>
116             <TapGestureRecognizer Tapped="NavigateToLocalProduct"/>

```

```
115         </StackLayout.GestureRecognizers>
116     </StackLayout>
117     <StackLayout Grid.Column="0"
118         Grid.Row="2"
119         VerticalOptions="End"
120         Spacing="5"
121         Margin="40,0,40,15">
122         <StackLayout.GestureRecognizers>
123             <TapGestureRecognizer Tapped="NavigateToLocalProduct"/>
124         </StackLayout.GestureRecognizers>
125         <Label Text="Buy Local"
126             TextColor="White"
127             HorizontalOptions="Center"
128             FontSize="24"
129             FontAttributes="Bold"
130             FontFamily="Proxima Nova"/>
131         <Label Text="Buy Local products and support the community."
132             TextColor="White"
133             HorizontalOptions="Center"
134             FontSize="15"
135             FontFamily="Proxima Nova Thin"
136             HorizontalTextAlignment="Center"/>
137     </StackLayout>
138
139     <Image Grid.Column="0"
140         Grid.Row="3"
141         Aspect="AspectFill"
142         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.OrganicFood.jpg}"/>
143     <StackLayout Grid.Column="0"
144         Grid.Row="3"
145         BackgroundColor="Black"
146         Opacity=".5">
147         <StackLayout.GestureRecognizers>
148             <TapGestureRecognizer Tapped="NavigateToOrganicFood"/>
149         </StackLayout.GestureRecognizers>
150     </StackLayout>
151     <StackLayout Grid.Column="0"
152         Grid.Row="3"
153         VerticalOptions="End"
154         Spacing="5"
155         Margin="40,0,40,15">
156         <StackLayout.GestureRecognizers>
157             <TapGestureRecognizer Tapped="NavigateToOrganicFood"/>
158         </StackLayout.GestureRecognizers>
159         <Label Text="Go Organic"
160             TextColor="White"
161             HorizontalOptions="Center"
162             FontSize="24"
163             FontAttributes="Bold"
164             FontFamily="Proxima Nova"/>
165         <Label Text="Purchase and make organic food over non organic"
166             TextColor="White"
167             HorizontalOptions="Center"
168             FontSize="15"
169             FontFamily="Proxima Nova Thin"
170             HorizontalTextAlignment="Center"/>
171     </StackLayout>
172
173     <Image Grid.Column="0"
174         Grid.Row="4"
```

```

175         Aspect="AspectFill"
176         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.FoodBulk.jpg}"/>
177     <StackLayout Grid.Column="0"
178         Grid.Row="4"
179         BackgroundColor="Black"
180         Opacity=".5">
181         <StackLayout.GestureRecognizers>
182             <TapGestureRecognizer Tapped="NavigateToFoodInBulk"/>
183         </StackLayout.GestureRecognizers>
184     </StackLayout>
185     <StackLayout Grid.Column="0"
186         Grid.Row="4"
187         VerticalOptions="End"
188         Spacing="5"
189         Margin="40,0,40,15">
190     <StackLayout.GestureRecognizers>
191         <TapGestureRecognizer Tapped="NavigateToFoodInBulk"/>
192     </StackLayout.GestureRecognizers>
193     <Label Text="Buy In Bulk"
194         TextColor="White"
195         HorizontalOptions="Center"
196         FontSize="24"
197         FontAttributes="Bold"
198         FontFamily="Proxima Nova"/>
199     <Label Text="Purchase food in bulk to save needless trips to the
shops."
200         TextColor="White"
201         HorizontalOptions="Center"
202         FontSize="15"
203         FontFamily="Proxima Nova Thin"
204         HorizontalTextAlignment="Center"/>
205 </StackLayout>
206
207     <Image Grid.Column="0"
208         Grid.Row="5"
209         Aspect="AspectFill"
210         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.EcoProduct.jpg}"/>
211     <StackLayout Grid.Column="0"
212         Grid.Row="5"
213         BackgroundColor="Black"
214         Opacity=".5">
215     <StackLayout.GestureRecognizers>
216         <TapGestureRecognizer Tapped="NavigateToEcoFriendlyProduct"/>
217     </StackLayout.GestureRecognizers>
218 </StackLayout>
219     <StackLayout Grid.Column="0"
220         Grid.Row="5"
221         VerticalOptions="End"
222         Spacing="5"
223         Margin="40,0,40,15">
224     <StackLayout.GestureRecognizers>
225         <TapGestureRecognizer Tapped="NavigateToEcoFriendlyProduct"/>
226     </StackLayout.GestureRecognizers>
227     <Label Text="Eco Product"
228         TextColor="White"
229         HorizontalOptions="Center"
230         FontSize="24"
231         FontAttributes="Bold"
232         FontFamily="Proxima Nova"/>
233     <Label Text="Purchase and Eco Friendly Product over a non Eco Friendly

```

```

Product."
234         TextColor="White"
235         HorizontalOptions="Center"
236         FontSize="15"
237         FontFamily="Proxima Nova Thin"
238         HorizontalTextAlignment="Center"/>
239     </StackLayout>
240
241     <Image Grid.Column="0"
242           Grid.Row="6"
243           Aspect="AspectFill"
244           Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.EthicalClothes.jpg}"/>
245     <StackLayout Grid.Column="0"
246                 Grid.Row="6"
247                 BackgroundColor="Black"
248                 Opacity=".5">
249         <StackLayout.GestureRecognizers>
250             <TapGestureRecognizer Tapped="NavigateToEthicalClothes"/>
251         </StackLayout.GestureRecognizers>
252     </StackLayout>
253     <StackLayout Grid.Column="0"
254                 Grid.Row="6"
255                 VerticalOptions="End"
256                 Spacing="5"
257                 Margin="40,0,40,15">
258         <StackLayout.GestureRecognizers>
259             <TapGestureRecognizer Tapped="NavigateToEthicalClothes"/>
260         </StackLayout.GestureRecognizers>
261         <Label Text="Purchase Ethical Clothes"
262               TextColor="White"
263               HorizontalOptions="Center"
264               FontSize="24"
265               FontAttributes="Bold"
266               FontFamily="Proxima Nova"/>
267         <Label Text="Instead of purchasing clothes that are not ethical
purchase Ethical Clothes."
268               TextColor="White"
269               HorizontalOptions="Center"
270               FontSize="15"
271               FontFamily="Proxima Nova Thin"
272               HorizontalTextAlignment="Center"/>
273     </StackLayout>
274
275     <Image Grid.Column="0"
276           Grid.Row="7"
277           Aspect="AspectFill"
278           Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.EcoBrush.jpg}"/>
279     <StackLayout Grid.Column="0"
280                 Grid.Row="7"
281                 BackgroundColor="Black"
282                 Opacity=".5">
283         <StackLayout.GestureRecognizers>
284             <TapGestureRecognizer Tapped="NavigateToEcoFriendlyToothbrush"/>
285         </StackLayout.GestureRecognizers>
286     </StackLayout>
287     <StackLayout Grid.Column="0"
288                 Grid.Row="7"
289                 VerticalOptions="End"
290                 Spacing="5"
291                 Margin="40,0,40,15">

```

```

292         <StackLayout.GestureRecognizers>
293             <TapGestureRecognizer Tapped="NavigateToEcoFriendlyToothbrush"/>
294         </StackLayout.GestureRecognizers>
295         <Label Text="Eco Toothbrush"
296             TextColor="White"
297             HorizontalOptions="Center"
298             FontSize="24"
299             FontAttributes="Bold"
300             FontFamily="Proxima Nova"/>
301         <Label Text="Purchase and use an Eco Friendly Toothbrush and an Eco
Friendly Toothbrush."
302             TextColor="White"
303             HorizontalOptions="Center"
304             FontSize="15"
305             FontFamily="Proxima Nova Thin"
306             HorizontalTextAlignment="Center"/>
307     </StackLayout>
308
309     <Image Grid.Column="0"
310         Grid.Row="8"
311         Aspect="AspectFill"
312         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.EcoAppliance.jpg}"/>
313     <StackLayout Grid.Column="0"
314         Grid.Row="8"
315         BackgroundColor="Black"
316         Opacity=".5">
317         <StackLayout.GestureRecognizers>
318             <TapGestureRecognizer Tapped="NavigateToEcoFreidnlyAppliance"/>
319         </StackLayout.GestureRecognizers>
320     </StackLayout>
321     <StackLayout Grid.Column="0"
322         Grid.Row="8"
323         VerticalOptions="End"
324         Spacing="5"
325         Margin="40,0,40,15">
326         <StackLayout.GestureRecognizers>
327             <TapGestureRecognizer Tapped="NavigateToEcoFreidnlyAppliance"/>
328         </StackLayout.GestureRecognizers>
329         <Label Text="Eco Appliance"
330             TextColor="White"
331             HorizontalOptions="Center"
332             FontSize="24"
333             FontAttributes="Bold"
334             FontFamily="Proxima Nova"/>
335         <Label Text="Purchase and use and Eco Friendly Appliance."
336             TextColor="White"
337             HorizontalOptions="Center"
338             FontSize="15"
339             FontFamily="Proxima Nova Thin"
340             HorizontalTextAlignment="Center"/>
341     </StackLayout>
342
343     <Image Grid.Column="0"
344         Grid.Row="9"
345         Aspect="AspectFill"
346         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.LooseTea.jpg}"/>
347     <StackLayout Grid.Column="0"
348         Grid.Row="9"
349         BackgroundColor="Black"
350         Opacity=".5">

```



```

351         <StackLayout.GestureRecognizers>
352             <TapGestureRecognizer Tapped="NavigateToLooseLeafTea"/>
353         </StackLayout.GestureRecognizers>
354     </StackLayout>
355     <StackLayout Grid.Column="0"
356         Grid.Row="9"
357         VerticalOptions="End"
358         Spacing="5"
359         Margin="40,0,40,15">
360         <StackLayout.GestureRecognizers>
361             <TapGestureRecognizer Tapped="NavigateToLooseLeafTea"/>
362         </StackLayout.GestureRecognizers>
363         <Label Text="Loose Leaf Tea"
364             TextColor="White"
365             HorizontalOptions="Center"
366             FontSize="24"
367             FontAttributes="Bold"
368             FontFamily="Proxima Nova"/>
369         <Label Text="No need for bagged tea. Just use Loose Leaf Tea instead."
370             TextColor="White"
371             HorizontalOptions="Center"
372             FontSize="15"
373             FontFamily="Proxima Nova Thin"
374             HorizontalTextAlignment="Center"/>
375     </StackLayout>
376
377     <Image Grid.Column="0"
378         Grid.Row="10"
379         Aspect="AspectFill"
380         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Energy.ReBatteries.jpg}"/>
381     <StackLayout Grid.Column="0"
382         Grid.Row="10"
383         BackgroundColor="Black"
384         Opacity=".5">
385         <StackLayout.GestureRecognizers>
386             <TapGestureRecognizer Tapped="NavigateToReBatteries"/>
387         </StackLayout.GestureRecognizers>
388     </StackLayout>
389     <StackLayout Grid.Column="0"
390         Grid.Row="10"
391         VerticalOptions="End"
392         Spacing="5"
393         Margin="40,0,40,15">
394         <StackLayout.GestureRecognizers>
395             <TapGestureRecognizer Tapped="NavigateToReBatteries"/>
396         </StackLayout.GestureRecognizers>
397         <Label Text="Reusable Batteries"
398             TextColor="White"
399             HorizontalOptions="Center"
400             FontSize="24"
401             FontAttributes="Bold"
402             FontFamily="Proxima Nova"/>
403         <Label Text="Purchase and use Rechargeable Batteries over non
rechargeable ones."
404             TextColor="White"
405             HorizontalOptions="Center"
406             FontSize="15"
407             FontFamily="Proxima Nova Thin"
408             HorizontalTextAlignment="Center"/>
409     </StackLayout>
410

```



```

411         <Image Grid.Column="0"
412             Grid.Row="11"
413             Aspect="AspectFill"
414             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.Napkin.jpg}"/>
415         <StackLayout Grid.Column="0"
416             Grid.Row="11"
417             BackgroundColor="Black"
418             Opacity=".5">
419             <StackLayout.GestureRecognizers>
420                 <TapGestureRecognizer Tapped="NavigateToClothNapkins"/>
421             </StackLayout.GestureRecognizers>
422         </StackLayout>
423         <StackLayout Grid.Column="0"
424             Grid.Row="11"
425             VerticalOptions="End"
426             Spacing="5"
427             Margin="40,0,40,15">
428             <StackLayout.GestureRecognizers>
429                 <TapGestureRecognizer Tapped="NavigateToClothNapkins"/>
430             </StackLayout.GestureRecognizers>
431             <Label Text="Cloth Napkins"
432                 TextColor="White"
433                 HorizontalOptions="Center"
434                 FontSize="24"
435                 FontAttributes="Bold"
436                 FontFamily="Proxima Nova"/>
437             <Label Text="Purchase and use Cloth Napkins over paper ones."
438                 TextColor="White"
439                 HorizontalOptions="Center"
440                 FontSize="15"
441                 FontFamily="Proxima Nova Thin"
442                 HorizontalTextAlignment="Center"/>
443         </StackLayout>
444
445         <Image Grid.Column="0"
446             Grid.Row="12"
447             Aspect="AspectFill"
448             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Shopping.Towel.jpg}"
449             Margin="0,0,0,5"/>
450         <StackLayout Grid.Column="0"
451             Grid.Row="12"
452             BackgroundColor="Black"
453             Opacity=".5"
454             Margin="0,0,0,5">
455             <StackLayout.GestureRecognizers>
456                 <TapGestureRecognizer Tapped="NavigateToClothTowels"/>
457             </StackLayout.GestureRecognizers>
458         </StackLayout>
459         <StackLayout Grid.Column="0"
460             Grid.Row="12"
461             VerticalOptions="End"
462             Spacing="5"
463             Margin="40,0,40,15">
464             <StackLayout.GestureRecognizers>
465                 <TapGestureRecognizer Tapped="NavigateToClothTowels"/>
466             </StackLayout.GestureRecognizers>
467             <Label Text="Cloth Towels"
468                 TextColor="White"
469                 HorizontalOptions="Center"

```

```
470         FontSize="24"  
471         FontAttributes="Bold"  
472         FontFamily="Proxima Nova"/>  
473     <Label Text="Purchase and use Cloth Towels over paper ones."  
474         TextColor="White"  
475         HorizontalOptions="Center"  
476         FontSize="15"  
477         FontFamily="Proxima Nova Thin"  
478         HorizontalTextAlignment="Center"/>  
479     </StackLayout>  
480 </Grid>  
481 </ScrollView>  
482 </ContentPage.Content>  
483 </ContentPage>
```

```
1 /!* \class The ShoppingPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the ShoppingPage View Class. This page displays and allows the
   navigation to each of the actions in the Shopping category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Shopping;
11 using System;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class ShoppingPage : ContentPage
19     {
20         public ShoppingPage()
21         {
22             InitializeComponent();
23             OnAppearing();
24         }
25         /** This function navigates to PurchaseReusableWater.
26         */
27         private async void NavigateToReusableWater(object sender, EventArgs e)
28         {
29             await Navigation.PushAsync(new PurchaseReusableWater());
30         }
31         /** This function navigates to ReusableBag.
32         */
33         private async void NavigateToPurchaseReusableBag(object sender, EventArgs e)
34         {
35             await Navigation.PushAsync(new ReusableBag());
36         }
37         /** This function navigates to LocalProduct.
38         */
39         private async void NavigateToLocalProduct(object sender, EventArgs e)
40         {
41             await Navigation.PushAsync(new LocalProduct());
42         }
43         /** This function navigates to OrganicFood.
44         */
45         private async void NavigateToOrganicFood(object sender, EventArgs e)
46         {
47             await Navigation.PushAsync(new OrganicFood());
48         }
49         /** This function navigates to FoodInBulk.
50         */
51         private async void NavigateToFoodInBulk(object sender, EventArgs e)
52         {
53             await Navigation.PushAsync(new FoodInBulk());
54         }
55         /** This function navigates to EcoFriendlyProduct.
56         */
57         private async void NavigateToEcoFriendlyProduct(object sender, EventArgs e)
58         {
59             await Navigation.PushAsync(new EcoFriendlyProduct());
```

```
60     }
61     /** This function navigates to EthicalClothes.
62     */
63     private async void NavigateToEthicalClothes(object sender, EventArgs e)
64     {
65         await Navigation.PushAsync(new EthicalClothes());
66     }
67     /** This function navigates to EcoFriendlyToothbrush.
68     */
69     private async void NavigateToEcoFriendlyToothbrush(object sender, EventArgs e)
70     {
71         await Navigation.PushAsync(new EcoFriendlyToothbrush());
72     }
73     /** This function navigates to EcoFreidnlyApplicance.
74     */
75     private async void NavigateToEcoFreidnlyApplicance(object sender, EventArgs e)
76     {
77         await Navigation.PushAsync(new EcoFreidnlyApplicance());
78     }
79     /** This function navigates to LooseLeafTea.
80     */
81     private async void NavigateToLooseLeafTea(object sender, EventArgs e)
82     {
83         await Navigation.PushAsync(new LooseLeafTea());
84     }
85     /** This function navigates to ReBatteries.
86     */
87     private async void NavigateToReBatteries(object sender, EventArgs e)
88     {
89         await Navigation.PushAsync(new ReBatteries());
90     }
91     /** This function navigates to ClothNapkins.
92     */
93     private async void NavigateToClothNapkins(object sender, EventArgs e)
94     {
95         await Navigation.PushAsync(new ClothNapkins());
96     }
97     /** This function navigates to ClothTowels.
98     */
99     private async void NavigateToClothTowels(object sender, EventArgs e)
100    {
101        await Navigation.PushAsync(new ClothTowels());
102    }
103    /** This function is called before the page is displayed and it created an object
104    ans uses it's SetLvl method to set the players level in the app
105    * and display it in the navigation bar.
106    */
107    protected override void OnAppearing()
108    {
109        GetData data = new GetData();
110        data.SetLvl();
111
112        theLevel.Text = "LVL: " + GetData.lvl.ToString();
113    }
114 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.TravelPage"
6     xmlns:local="clr-namespace:Application_Green_Quake.Models"
7     Title="Travel Subcategories">
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Travel" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15            Margin="0,0,30,0"/>
16        </StackLayout>
17    </NavigationPage.TitleView>
18
19    <ContentPage.Content>
20        <ScrollView>
21            <Grid Margin="5,5,5,5">
22                <Grid.RowDefinitions>
23                    <RowDefinition Height="200" />
24                    <RowDefinition Height="200" />
25                    <RowDefinition Height="200" />
26                    <RowDefinition Height="200" />
27                    <RowDefinition Height="200" />
28                </Grid.RowDefinitions>
29                <Grid.ColumnDefinitions>
30                    <ColumnDefinition />
31                </Grid.ColumnDefinitions>
32
33                <Image Grid.Column="0"
34                    Grid.Row="0"
35                    Aspect="AspectFill"
36                    Source="{local:ImageResource
37        Application_Green_Quake.Images.SubCategories.Travel.Carpool.jpg}"/>
38                <StackLayout Grid.Column="0"
39                    Grid.Row="0"
40                    BackgroundColor="Black"
41                    Opacity=".5">
42                    <StackLayout.GestureRecognizers>
43                        <TapGestureRecognizer Tapped="NavigateToCarpool"/>
44                    </StackLayout.GestureRecognizers>
45                </StackLayout>
46                <StackLayout Grid.Column="0"
47                    Grid.Row="0"
48                    VerticalOptions="End"
49                    Spacing="5"
50                    Margin="40,0,40,15">
51                    <StackLayout.GestureRecognizers>
52                        <TapGestureRecognizer Tapped="NavigateToCarpool"/>
53                    </StackLayout.GestureRecognizers>
54                    <Label Text="Carpool"
55                        TextColor="White"
56                        HorizontalOptions="Center"
57                        FontSize="24"
58                        FontAttributes="Bold"
59                        FontFamily="Proxima Nova"/>
60                    <Label Text="Carpool when traveling to work or school."
61                        TextColor="White"

```

```
57         HorizontalOptions="Center"
58         FontSize="15"
59         FontFamily="Proxima Nova Thin"
60         HorizontalTextAlignment="Center"/>
61     </StackLayout>
62
63     <Image Grid.Column="0"
64           Grid.Row="1"
65           Aspect="AspectFill"
66           Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.PublicTransport.jpg}"/>
67     <StackLayout Grid.Column="0"
68                 Grid.Row="1"
69                 BackgroundColor="Black"
70                 Opacity=".5">
71         <StackLayout.GestureRecognizers>
72             <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
73         </StackLayout.GestureRecognizers>
74     </StackLayout>
75     <StackLayout Grid.Column="0"
76                 Grid.Row="1"
77                 VerticalOptions="End"
78                 Spacing="5"
79                 Margin="40,0,40,15">
80         <StackLayout.GestureRecognizers>
81             <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
82         </StackLayout.GestureRecognizers>
83         <Label Text="Public Transport"
84               TextColor="White"
85               HorizontalOptions="Center"
86               FontSize="24"
87               FontAttributes="Bold"
88               FontFamily="Proxima Nova"/>
89         <Label Text="Use Public Transport over a car."
90               TextColor="White"
91               HorizontalOptions="Center"
92               FontSize="15"
93               FontFamily="Proxima Nova Thin"
94               HorizontalTextAlignment="Center"/>
95     </StackLayout>
96
97     <Image Grid.Column="0"
98           Grid.Row="2"
99           Aspect="AspectFill"
100          Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Walk.jpg}"/>
101     <StackLayout Grid.Column="0"
102                 Grid.Row="2"
103                 BackgroundColor="Black"
104                 Opacity=".5">
105         <StackLayout.GestureRecognizers>
106             <TapGestureRecognizer Tapped="NavigateToWalk"/>
107         </StackLayout.GestureRecognizers>
108     </StackLayout>
109     <StackLayout Grid.Column="0"
110                 Grid.Row="2"
111                 VerticalOptions="End"
112                 Spacing="5"
113                 Margin="40,0,40,15">
114         <StackLayout.GestureRecognizers>
115             <TapGestureRecognizer Tapped="NavigateToWalk"/>
116         </StackLayout.GestureRecognizers>
```

```

117         <Label Text="Walk"
118             TextColor="White"
119             HorizontalOptions="Center"
120             FontSize="24"
121             FontAttributes="Bold"
122             FontFamily="Proxima Nova"/>
123         <Label Text="Walk there if you can."
124             TextColor="White"
125             HorizontalOptions="Center"
126             FontSize="15"
127             FontFamily="Proxima Nova Thin"
128             HorizontalTextAlignment="Center"/>
129     </StackLayout>
130
131     <Image Grid.Column="0"
132         Grid.Row="3"
133         Aspect="AspectFill"
134         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Cycle.jpg}"/>
135     <StackLayout Grid.Column="0"
136         Grid.Row="3"
137         BackgroundColor="Black"
138         Opacity=".5">
139         <StackLayout.GestureRecognizers>
140             <TapGestureRecognizer Tapped="NavigateToCycle"/>
141         </StackLayout.GestureRecognizers>
142     </StackLayout>
143     <StackLayout Grid.Column="0"
144         Grid.Row="3"
145         VerticalOptions="End"
146         Spacing="5"
147         Margin="40,0,40,15">
148         <StackLayout.GestureRecognizers>
149             <TapGestureRecognizer Tapped="NavigateToCycle"/>
150         </StackLayout.GestureRecognizers>
151         <Label Text="Cycle"
152             TextColor="White"
153             HorizontalOptions="Center"
154             FontSize="24"
155             FontAttributes="Bold"
156             FontFamily="Proxima Nova"/>
157         <Label Text="Cycle instead if you can."
158             TextColor="White"
159             HorizontalOptions="Center"
160             FontSize="15"
161             FontFamily="Proxima Nova Thin"
162             HorizontalTextAlignment="Center"/>
163     </StackLayout>
164
165     <Image Grid.Column="0"
166         Grid.Row="4"
167         Aspect="AspectFill"
168         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.EcoCar.jpg}"
169         Margin="0,0,0,5"/>
170     <StackLayout Grid.Column="0"
171         Grid.Row="4"
172         BackgroundColor="Black"
173         Opacity=".5"
174         Margin="0,0,0,5">
175         <StackLayout.GestureRecognizers>

```

```
176         <TapGestureRecognizer Tapped="NavigateToEcoFreindlyCar"/>
177     </StackLayout.GestureRecognizers>
178 </StackLayout>
179 <StackLayout Grid.Column="0"
180     Grid.Row="4"
181     VerticalOptions="End"
182     Spacing="5"
183     Margin="40,0,40,15">
184     <StackLayout.GestureRecognizers>
185         <TapGestureRecognizer Tapped="NavigateToEcoFreindlyCar"/>
186     </StackLayout.GestureRecognizers>
187     <Label Text="Eco Car"
188         TextColor="White"
189         HorizontalOptions="Center"
190         FontSize="24"
191         FontAttributes="Bold"
192         FontFamily="Proxima Nova"/>
193     <Label Text="Purchase and use and Eco Friendly Car."
194         TextColor="White"
195         HorizontalOptions="Center"
196         FontSize="15"
197         FontFamily="Proxima Nova Thin"
198         HorizontalTextAlignment="Center"/>
199 </StackLayout>
200 </Grid>
201 </ScrollView>
202 </ContentPage.Content>
203 </ContentPage>
```



```

1  /!* \class The TravelPage View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the TravelPage View Class. This page displays and allows the
   navigation to each of the actions in the Travel category.
7  *
8  */
9  using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Travel;
11 using System;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class TravelPage : ContentPage
19     {
20         public TravelPage()
21         {
22             InitializeComponent();
23             OnAppearing();
24         }
25         /** This function navigates to Carpool.
26         */
27         private async void NavigateToCarpool(object sender, EventArgs e)
28         {
29             await Navigation.PushAsync(new Carpool());
30         }
31         /** This function navigates to PublicTransport.
32         */
33         private async void NavigateToPublicTransport(object sender, EventArgs e)
34         {
35             await Navigation.PushAsync(new PublicTransport());
36         }
37         /** This function navigates to Walk.
38         */
39         private async void NavigateToWalk(object sender, EventArgs e)
40         {
41             await Navigation.PushAsync(new Walk());
42         }
43         /** This function navigates to Cycle.
44         */
45         private async void NavigateToCycle(object sender, EventArgs e)
46         {
47             await Navigation.PushAsync(new Cycle());
48         }
49         /** This function navigates to EcoFreindlyCar.
50         */
51         private async void NavigateToEcoFreindlyCar(object sender, EventArgs e)
52         {
53             await Navigation.PushAsync(new EcoFreindlyCar());
54         }
55         /** 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
56         * and display it in the navigation bar.
57         */
58         protected override void OnAppearing()

```

```
59 |     {
60 |         GetData data = new GetData();
61 |         data.SetLvl();
62 |
63 |         theLevel.Text = "LVL: " + GetData.lvl.ToString();
64 |     }
65 | }
66 | }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.WastePage"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models"
6     Title="Waste Subcategories">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Waste" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15            Margin="0,0,30,0"/>
16         </StackLayout>
17     </NavigationPage.TitleView>
18
19     <ContentPage.Content>
20         <ScrollView>
21             <Grid Margin="5,5,5,5">
22                 <Grid.RowDefinitions>
23                     <RowDefinition Height="200" />
24                     <RowDefinition Height="200" />
25                     <RowDefinition Height="200" />
26                     <RowDefinition Height="200" />
27                     <RowDefinition Height="200" />
28                 </Grid.RowDefinitions>
29                 <Grid.ColumnDefinitions>
30                     <ColumnDefinition />
31                 </Grid.ColumnDefinitions>
32
33                 <Image Grid.Column="0"
34                    Grid.Row="0"
35                    Aspect="AspectFill"
36                    Source="{local:ImageResource
37        Application_Green_Quake.Images.SubCategories.Waste.Recycled.jpg}"/>
38                 <StackLayout Grid.Column="0"
39                    Grid.Row="0"
40                    BackgroundColor="Black"
41                    Opacity=".5">
42                     <StackLayout.GestureRecognizers>
43                         <TapGestureRecognizer Tapped="NavigateToUseRecyclingBin"/>
44                     </StackLayout.GestureRecognizers>
45                 </StackLayout>
46                 <StackLayout Grid.Column="0"
47                    Grid.Row="0"
48                    VerticalOptions="End"
49                    Spacing="5"
50                    Margin="40,0,40,15">
51                     <StackLayout.GestureRecognizers>
52                         <TapGestureRecognizer Tapped="NavigateToUseRecyclingBin"/>
53                     </StackLayout.GestureRecognizers>
54                     <Label Text="Recycle"
55                        TextColor="White"
56                        HorizontalOptions="Center"
57                        FontSize="24"
58                        FontAttributes="Bold"
59                        FontFamily="Proxima Nova"/>
60                     <Label Text="Recycle your waste."
61                        TextColor="White"
62                        HorizontalOptions="Center"
```

```

58         FontSize="15"
59         FontFamily="Proxima Nova Thin"
60         HorizontalTextAlignment="Center"/>
61     </StackLayout>
62
63     <Image Grid.Column="0"
64           Grid.Row="1"
65           Aspect="AspectFill"
66           Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Waste.SetupBins.jpg}"/>
67     <StackLayout Grid.Column="0"
68           Grid.Row="1"
69           BackgroundColor="Black"
70           Opacity=".5">
71         <StackLayout.GestureRecognizers>
72             <TapGestureRecognizer Tapped="NavigateToSetUpRecyclingBin"/>
73         </StackLayout.GestureRecognizers>
74     </StackLayout>
75     <StackLayout Grid.Column="0"
76           Grid.Row="1"
77           VerticalOptions="End"
78           Spacing="5"
79           Margin="40,0,40,15">
80         <StackLayout.GestureRecognizers>
81             <TapGestureRecognizer Tapped="NavigateToSetUpRecyclingBin"/>
82         </StackLayout.GestureRecognizers>
83         <Label Text="Set Up Recycling Bins"
84               TextColor="White"
85               HorizontalOptions="Center"
86               FontSize="24"
87               FontAttributes="Bold"
88               FontFamily="Proxima Nova"/>
89         <Label Text="Set up recycling bins so you can begin you recycling
journey."
90               TextColor="White"
91               HorizontalOptions="Center"
92               FontSize="15"
93               FontFamily="Proxima Nova Thin"
94               HorizontalTextAlignment="Center"/>
95     </StackLayout>
96
97     <Image Grid.Column="0"
98           Grid.Row="2"
99           Aspect="AspectFill"
100          Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Waste.Compost.jpg}"/>
101     <StackLayout Grid.Column="0"
102           Grid.Row="2"
103           BackgroundColor="Black"
104           Opacity=".5">
105         <StackLayout.GestureRecognizers>
106             <TapGestureRecognizer Tapped="NavigateToSetUpCompostBin"/>
107         </StackLayout.GestureRecognizers>
108     </StackLayout>
109     <StackLayout Grid.Column="0"
110           Grid.Row="2"
111           VerticalOptions="End"
112           Spacing="5"
113           Margin="40,0,40,15">
114         <StackLayout.GestureRecognizers>
115             <TapGestureRecognizer Tapped="NavigateToSetUpCompostBin"/>
116     </StackLayout.GestureRecognizers>

```

```

117         <Label Text="Set Up Compost Bins"
118             TextColor="White"
119             HorizontalOptions="Center"
120             FontSize="24"
121             FontAttributes="Bold"
122             FontFamily="Proxima Nova"/>
123         <Label Text="Set up compost bins so you can start composting your food
waste."
124             TextColor="White"
125             HorizontalOptions="Center"
126             FontSize="15"
127             FontFamily="Proxima Nova Thin"
128             HorizontalTextAlignment="Center"/>
129     </StackLayout>
130
131     <Image Grid.Column="0"
132         Grid.Row="3"
133         Aspect="AspectFill"
134         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Waste.BioBags.jpg}"/>
135     <StackLayout Grid.Column="0"
136         Grid.Row="3"
137         BackgroundColor="Black"
138         Opacity=".5">
139         <StackLayout.GestureRecognizers>
140             <TapGestureRecognizer Tapped="NavigateToUseBiogradableBinBags"/>
141         </StackLayout.GestureRecognizers>
142     </StackLayout>
143     <StackLayout Grid.Column="0"
144         Grid.Row="3"
145         VerticalOptions="End"
146         Spacing="5"
147         Margin="40,0,40,15">
148     <StackLayout.GestureRecognizers>
149         <TapGestureRecognizer Tapped="NavigateToUseBiogradableBinBags"/>
150     </StackLayout.GestureRecognizers>
151     <Label Text="Bio Bin Bags"
152         TextColor="White"
153         HorizontalOptions="Center"
154         FontSize="24"
155         FontAttributes="Bold"
156         FontFamily="Proxima Nova"/>
157     <Label Text="Purchase and use Biodegradable Bin Bags instead."
158         TextColor="White"
159         HorizontalOptions="Center"
160         FontSize="15"
161         FontFamily="Proxima Nova Thin"
162         HorizontalTextAlignment="Center"/>
163 </StackLayout>
164
165     <Image Grid.Column="0"
166         Grid.Row="4"
167         Aspect="AspectFill"
168         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Waste.OnlineBills.jpg}"
169         Margin="0,0,0,5"/>
170     <StackLayout Grid.Column="0"
171         Grid.Row="4"
172         BackgroundColor="Black"
173         Opacity=".5"
174         Margin="0,0,0,5">
175     <StackLayout.GestureRecognizers>

```

```
176         <TapGestureRecognizer Tapped="NavigateToPayBillsOnline"/>
177     </StackLayout.GestureRecognizers>
178 </StackLayout>
179 <StackLayout Grid.Column="0"
180     Grid.Row="4"
181     VerticalOptions="End"
182     Spacing="5"
183     Margin="40,0,40,15">
184     <StackLayout.GestureRecognizers>
185         <TapGestureRecognizer Tapped="NavigateToPayBillsOnline"/>
186     </StackLayout.GestureRecognizers>
187     <Label Text="Pay Bills Online"
188         TextColor="White"
189         HorizontalOptions="Center"
190         FontSize="24"
191         FontAttributes="Bold"
192         FontFamily="Proxima Nova"/>
193     <Label Text="Pay your Bills online. No need to waste paper.."
194         TextColor="White"
195         HorizontalOptions="Center"
196         FontSize="15"
197         FontFamily="Proxima Nova Thin"
198         HorizontalTextAlignment="Center"/>
199 </StackLayout>
200 </Grid>
201 </ScrollView>
202 </ContentPage.Content>
203 </ContentPage>
```

```

1  /!* \class The WastePage View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the WastePage View Class. This page displays and allows the
   navigation to each of the actions in the Waste category.
7  *
8  */
9  using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Waste;
11 using System;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class WastePage : ContentPage
19     {
20         public WastePage()
21         {
22             InitializeComponent();
23             OnAppearing();
24         }
25         /** This function navigates to UseRecyclingBin.
26         */
27         private async void NavigateToUseRecyclingBin(object sender, EventArgs e)
28         {
29             await Navigation.PushAsync(new UseRecyclingBin());
30         }
31         /** This function navigates to SetUpRecyclingBin.
32         */
33         private async void NavigateToSetUpRecyclingBin(object sender, EventArgs e)
34         {
35             await Navigation.PushAsync(new SetUpRecyclingBin());
36         }
37         /** This function navigates to CompostWaste.
38         */
39         private async void NavigateToSetUpCompostBin(object sender, EventArgs e)
40         {
41             await Navigation.PushAsync(new CompostWaste());
42         }
43         /** This function navigates to UseBiogradableBinBags.
44         */
45         private async void NavigateToUseBiogradableBinBags(object sender, EventArgs e)
46         {
47             await Navigation.PushAsync(new UseBiogradableBinBags());
48         }
49         /** This function navigates to BillsOnline.
50         */
51         private async void NavigateToPayBillsOnline(object sender, EventArgs e)
52         {
53             await Navigation.PushAsync(new BillsOnline());
54         }
55         /** 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
56         * and display it in the navigation bar.
57         */
58         protected override void OnAppearing()

```

```
59 |     {
60 |         GetData data = new GetData();
61 |         data.SetLvl();
62 |
63 |         theLevel.Text = "LVL: " + GetData.lvl.ToString();
64 |     }
65 | }
66 | }
```



```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.WaterPage"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models"
6     Title="Water Subcategories">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Water" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15            Margin="0,0,30,0"/>
16         </StackLayout>
17
18     </NavigationPage.TitleView>
19     <ContentPage.Content>
20         <ScrollView>
21             <Grid Margin="5,5,5,5">
22                 <Grid.RowDefinitions>
23                     <RowDefinition Height="200" />
24                     <RowDefinition Height="200" />
25                     <RowDefinition Height="200" />
26                     <RowDefinition Height="200" />
27                     <RowDefinition Height="200" />
28                     <RowDefinition Height="200" />
29                     <RowDefinition Height="200" />
30                     <RowDefinition Height="200" />
31                     <RowDefinition Height="200" />
32                     <RowDefinition Height="200" />
33                 </Grid.RowDefinitions>
34                 <Grid.ColumnDefinitions>
35                     <ColumnDefinition />
36                 </Grid.ColumnDefinitions>
37
38                 <Image Grid.Column="0"
39                    Grid.Row="0"
40                    Aspect="AspectFill"
41                    Source="{local:ImageResource
42        Application_Green_Quake.Images.SubCategories.Habits.TimedBrushing.jpg}"/>
43                 <StackLayout Grid.Column="0"
44                    Grid.Row="0"
45                    BackgroundColor="Black"
46                    Opacity=".5">
47                     <StackLayout.GestureRecognizers>
48                         <TapGestureRecognizer Tapped="NavigateToBrushingPage"/>
49                     </StackLayout.GestureRecognizers>
50                 </StackLayout>
51                 <StackLayout Grid.Column="0"
52                    Grid.Row="0"
53                    VerticalOptions="End"
54                    Spacing="5"
55                    Margin="40,0,40,15">
56                     <StackLayout.GestureRecognizers>
57                         <TapGestureRecognizer Tapped="NavigateToBrushingPage"/>
58                     </StackLayout.GestureRecognizers>
59                     <Label Text="Tap Off"
60                        TextColor="White"
61                        HorizontalOptions="Center"
62                        FontSize="24"

```

```

58         FontAttributes="Bold"
59         FontFamily="Proxima Nova"/>
60     <Label Text="Turn off the tap when you are brushing your teeth and save
water and the planet."
61         TextColor="White"
62         HorizontalOptions="Center"
63         FontSize="15"
64         FontFamily="Proxima Nova Thin"
65         HorizontalTextAlignment="Center"/>
66     </StackLayout>
67
68     <Image Grid.Column="0"
69         Grid.Row="1"
70         Aspect="AspectFill"
71         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.TimedShower.jpg}"/>
72     <StackLayout Grid.Column="0"
73         Grid.Row="1"
74         BackgroundColor="Black"
75         Opacity=".5">
76         <StackLayout.GestureRecognizers>
77             <TapGestureRecognizer Tapped="NavigateToTimedShowerPage"/>
78         </StackLayout.GestureRecognizers>
79     </StackLayout>
80     <StackLayout Grid.Column="0"
81         Grid.Row="1"
82         VerticalOptions="End"
83         Spacing="5"
84         Margin="40,0,40,15">
85         <StackLayout.GestureRecognizers>
86             <TapGestureRecognizer Tapped="NavigateToTimedShowerPage"/>
87         </StackLayout.GestureRecognizers>
88         <Label Text="Time Your Shower"
89             TextColor="White"
90             HorizontalOptions="Center"
91             FontSize="24"
92             FontAttributes="Bold"
93             FontFamily="Proxima Nova"/>
94         <Label Text="Time your showers so they don't take too long and save
water."
95             TextColor="White"
96             HorizontalOptions="Center"
97             FontSize="15"
98             FontFamily="Proxima Nova Thin"
99             HorizontalTextAlignment="Center"/>
100     </StackLayout>
101
102     <Image Grid.Column="0"
103         Grid.Row="2"
104         Aspect="AspectFill"
105         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.ShowerNoBath.jpg}"/>
106     <StackLayout Grid.Column="0"
107         Grid.Row="2"
108         BackgroundColor="Black"
109         Opacity=".5">
110         <StackLayout.GestureRecognizers>
111             <TapGestureRecognizer Tapped="NavigateToShoweredInsteadOfBath"/>
112         </StackLayout.GestureRecognizers>
113     </StackLayout>
114     <StackLayout Grid.Column="0"
115         Grid.Row="2"

```

```

116         VerticalOptions="End"
117         Spacing="5"
118         Margin="40,0,40,15">
119     <StackLayout.GestureRecognizers>
120         <TapGestureRecognizer Tapped="NavigateToShoweredInsteadOfBath"/>
121     </StackLayout.GestureRecognizers>
122     <Label Text="Shower No Bath"
123         TextColor="White"
124         HorizontalOptions="Center"
125         FontSize="24"
126         FontAttributes="Bold"
127         FontFamily="Proxima Nova"/>
128     <Label Text="Take a brief shower over taking a bath."
129         TextColor="White"
130         HorizontalOptions="Center"
131         FontSize="15"
132         FontFamily="Proxima Nova Thin"
133         HorizontalTextAlignment="Center"/>
134 </StackLayout>
135
136     <Image Grid.Column="0"
137         Grid.Row="3"
138         Aspect="AspectFill"
139         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.FullDishwasher.jpg}"/>
140     <StackLayout Grid.Column="0"
141         Grid.Row="3"
142         BackgroundColor="Black"
143         Opacity=".5">
144         <StackLayout.GestureRecognizers>
145             <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
146         </StackLayout.GestureRecognizers>
147     </StackLayout>
148     <StackLayout Grid.Column="0"
149         Grid.Row="3"
150         VerticalOptions="End"
151         Spacing="5"
152         Margin="40,0,40,15">
153     <StackLayout.GestureRecognizers>
154         <TapGestureRecognizer Tapped="NavigateToDishwasherFull"/>
155     </StackLayout.GestureRecognizers>
156     <Label Text="Full Dishwasher"
157         TextColor="White"
158         HorizontalOptions="Center"
159         FontSize="24"
160         FontAttributes="Bold"
161         FontFamily="Proxima Nova"/>
162     <Label Text="Only use the dishwasher when it is full."
163         TextColor="White"
164         HorizontalOptions="Center"
165         FontSize="15"
166         FontFamily="Proxima Nova Thin"
167         HorizontalTextAlignment="Center"/>
168 </StackLayout>
169
170     <Image Grid.Column="0"
171         Grid.Row="4"
172         Aspect="AspectFill"
173         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.FD.ReBottle.jpg}"/>
174     <StackLayout Grid.Column="0"
175         Grid.Row="4"

```

```

176         BackgroundColor="Black"
177         Opacity=".5">
178     <StackLayout.GestureRecognizers>
179         <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
180     </StackLayout.GestureRecognizers>
181 </StackLayout>
182 <StackLayout Grid.Column="0"
183     Grid.Row="4"
184     VerticalOptions="End"
185     Spacing="5"
186     Margin="40,0,40,15">
187     <StackLayout.GestureRecognizers>
188         <TapGestureRecognizer Tapped="NavigateToReusableWater"/>
189     </StackLayout.GestureRecognizers>
190     <Label Text="Use A Reusable Bottle"
191         TextColor="White"
192         HorizontalOptions="Center"
193         FontSize="24"
194         FontAttributes="Bold"
195         FontFamily="Proxima Nova"/>
196     <Label Text="Purchase and use a reusable water bottle instead of using
plastic bottles."
197         TextColor="White"
198         HorizontalOptions="Center"
199         FontSize="15"
200         FontFamily="Proxima Nova Thin"
201         HorizontalTextAlignment="Center"/>
202 </StackLayout>
203
204     <Image Grid.Column="0"
205         Grid.Row="5"
206         Aspect="AspectFill"
207         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Water.ShowerHead.jpg}"/>
208     <StackLayout Grid.Column="0"
209         Grid.Row="5"
210         BackgroundColor="Black"
211         Opacity=".5">
212     <StackLayout.GestureRecognizers>
213         <TapGestureRecognizer Tapped="NavigateToWSShowerhead"/>
214     </StackLayout.GestureRecognizers>
215 </StackLayout>
216 <StackLayout Grid.Column="0"
217     Grid.Row="5"
218     VerticalOptions="End"
219     Spacing="5"
220     Margin="40,0,40,15">
221     <StackLayout.GestureRecognizers>
222         <TapGestureRecognizer Tapped="NavigateToWSShowerhead"/>
223     </StackLayout.GestureRecognizers>
224     <Label Text="Water Saving Shower Head"
225         TextColor="White"
226         HorizontalOptions="Center"
227         FontSize="24"
228         FontAttributes="Bold"
229         FontFamily="Proxima Nova"/>
230     <Label Text="Install a Water Saving Shower Head to use less water."
231         TextColor="White"
232         HorizontalOptions="Center"
233         FontSize="15"
234         FontFamily="Proxima Nova Thin"

```

```

235         HorizontalTextAlignment="Center"/>
236     </StackLayout>
237
238     <Image Grid.Column="0"
239         Grid.Row="6"
240         Aspect="AspectFill"
241         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Water.CisternDis.jpg}"/>
242     <StackLayout Grid.Column="0"
243         Grid.Row="6"
244         BackgroundColor="Black"
245         Opacity=".5">
246         <StackLayout.GestureRecognizers>
247             <TapGestureRecognizer Tapped="NavigateToCisternDisplacement"/>
248         </StackLayout.GestureRecognizers>
249     </StackLayout>
250     <StackLayout Grid.Column="0"
251         Grid.Row="6"
252         VerticalOptions="End"
253         Spacing="5"
254         Margin="40,0,40,15">
255         <StackLayout.GestureRecognizers>
256             <TapGestureRecognizer Tapped="NavigateToCisternDisplacement"/>
257         </StackLayout.GestureRecognizers>
258         <Label Text="Cistern Displacement System"
259             TextColor="White"
260             HorizontalOptions="Center"
261             FontSize="24"
262             FontAttributes="Bold"
263             FontFamily="Proxima Nova"/>
264         <Label Text="Install a Cistern Displacement System to save more water."
265             TextColor="White"
266             HorizontalOptions="Center"
267             FontSize="15"
268             FontFamily="Proxima Nova Thin"
269             HorizontalTextAlignment="Center"/>
270     </StackLayout>
271
272     <Image Grid.Column="0"
273         Grid.Row="7"
274         Aspect="AspectFill"
275         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Water.ShowerBucket.jpg}"/>
276     <StackLayout Grid.Column="0"
277         Grid.Row="7"
278         BackgroundColor="Black"
279         Opacity=".5">
280         <StackLayout.GestureRecognizers>
281             <TapGestureRecognizer Tapped="NavigateToShowerBucket"/>
282         </StackLayout.GestureRecognizers>
283     </StackLayout>
284     <StackLayout Grid.Column="0"
285         Grid.Row="7"
286         VerticalOptions="End"
287         Spacing="5"
288         Margin="40,0,40,15">
289         <StackLayout.GestureRecognizers>
290             <TapGestureRecognizer Tapped="NavigateToShowerBucket"/>
291         </StackLayout.GestureRecognizers>
292         <Label Text="Shower Bucket"
293             TextColor="White"
294             HorizontalOptions="Center"

```

```
295         FontSize="24"
296         FontAttributes="Bold"
297         FontFamily="Proxima Nova"/>
298         <Label Text="Collect the water when you shower and use it for something
else."
299         TextColor="White"
300         HorizontalOptions="Center"
301         FontSize="15"
302         FontFamily="Proxima Nova Thin"
303         HorizontalTextAlignment="Center"/>
304     </StackLayout>
305
306     <Image Grid.Column="0"
307         Grid.Row="8"
308         Aspect="AspectFill"
309         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Water.RainBarrel.jpg}"/>
310     <StackLayout Grid.Column="0"
311         Grid.Row="8"
312         BackgroundColor="Black"
313         Opacity=".5">
314         <StackLayout.GestureRecognizers>
315             <TapGestureRecognizer Tapped="NavigateToRainBarrel"/>
316         </StackLayout.GestureRecognizers>
317     </StackLayout>
318     <StackLayout Grid.Column="0"
319         Grid.Row="8"
320         VerticalOptions="End"
321         Spacing="5"
322         Margin="40,0,40,15">
323         <StackLayout.GestureRecognizers>
324             <TapGestureRecognizer Tapped="NavigateToRainBarrel"/>
325         </StackLayout.GestureRecognizers>
326         <Label Text="Rain Barrel"
327             TextColor="White"
328             HorizontalOptions="Center"
329             FontSize="24"
330             FontAttributes="Bold"
331             FontFamily="Proxima Nova"/>
332         <Label Text="Set up a Rain Barrel to collect rainwater which can be
reused for something else.."
333             TextColor="White"
334             HorizontalOptions="Center"
335             FontSize="15"
336             FontFamily="Proxima Nova Thin"
337             HorizontalTextAlignment="Center"/>
338     </StackLayout>
339
340     <Image Grid.Column="0"
341         Grid.Row="9"
342         Aspect="AspectFill"
343         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Home.Flush.jpg}"
344         Margin="0,0,0,5"/>
345     <StackLayout Grid.Column="0"
346         Grid.Row="9"
347         BackgroundColor="Black"
348         Opacity=".5"
349         Margin="0,0,0,5">
350         <StackLayout.GestureRecognizers>
351             <TapGestureRecognizer Tapped="NavigateToToiletFlushes"/>
352     </StackLayout.GestureRecognizers>
```

```
353         </StackLayout>
354         <StackLayout Grid.Column="0"
355             Grid.Row="9"
356             VerticalOptions="End"
357             Spacing="5"
358             Margin="40,0,40,15">
359             <StackLayout.GestureRecognizers>
360                 <TapGestureRecognizer Tapped="NavigateToToiletFlushes"/>
361             </StackLayout.GestureRecognizers>
362             <Label Text="Save A Flush"
363                 TextColor="White"
364                 HorizontalOptions="Center"
365                 FontSize="24"
366                 FontAttributes="Bold"
367                 FontFamily="Proxima Nova"/>
368             <Label Text="Save a flush when you can and save water."
369                 TextColor="White"
370                 HorizontalOptions="Center"
371                 FontSize="15"
372                 FontFamily="Proxima Nova Thin"
373                 HorizontalTextAlignment="Center"/>
374         </StackLayout>
375     </Grid>
376 </ScrollView>
377 </ContentPage.Content>
378 </ContentPage>
```



```
1 /!* \class The WaterPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the WaterPage View Class. This page displays and allows the
   navigation to each of the actions in the Water category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Habits;
11 using Application_Green_Quake.Views.EcoActions.Home;
12 using Application_Green_Quake.Views.EcoActions.Water;
13 using System;
14 using Xamarin.Forms;
15 using Xamarin.Forms.Xaml;
16
17 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
18 {
19     [XamlCompilation(XamlCompilationOptions.Compile)]
20     public partial class WaterPage : ContentPage
21     {
22         public WaterPage()
23         {
24             InitializeComponent();
25             OnAppearing();
26         }
27         /** This function navigates to BrushingTeeth.
28         */
29         private async void NavigateToBrushingPage(object sender, EventArgs e)
30         {
31             await Navigation.PushAsync(new BrushingTeeth());
32         }
33         /** This function navigates to TimedShower.
34         */
35         private async void NavigateToTimedShowerPage(object sender, EventArgs e)
36         {
37             await Navigation.PushAsync(new TimedShower());
38         }
39         /** This function navigates to ShowerInstead.
40         */
41         private async void NavigateToShoweredInsteadOfBath(object sender, EventArgs e)
42         {
43             await Navigation.PushAsync(new ShowerInstead());
44         }
45         /** This function navigates to DishwasherFull.
46         */
47         private async void NavigateToDishwasherFull(object sender, EventArgs e)
48         {
49             await Navigation.PushAsync(new DishwasherFull());
50         }
51         /** This function navigates to ReusableWater.
52         */
53         private async void NavigateToReusableWater(object sender, EventArgs e)
54         {
55             await Navigation.PushAsync(new ReusableWater());
56         }
57         /** This function navigates to WSShowerHead.
58         */
59     }
```



```
60 private async void NavigateToWSShowerhead(object sender, EventArgs e)
61 {
62     await Navigation.PushAsync(new WSShowerHead());
63 }
64 /** This function navigates to CisternDisplacement.
65 */
66 private async void NavigateToCisternDisplacement(object sender, EventArgs e)
67 {
68     await Navigation.PushAsync(new CisternDisplacement());
69 }
70 /** This function navigates to ShowerBucket.
71 */
72 private async void NavigateToShowerBucket(object sender, EventArgs e)
73 {
74     await Navigation.PushAsync(new ShowerBucket());
75 }
76 /** This function navigates to RainBarrel.
77 */
78 private async void NavigateToRainBarrel(object sender, EventArgs e)
79 {
80     await Navigation.PushAsync(new RainBarrel());
81 }
82 /** This function navigates to ToiletFlushes.
83 */
84 private async void NavigateToToiletFlushes(object sender, EventArgs e)
85 {
86     await Navigation.PushAsync(new ToiletFlushes());
87 }
88 /** This function is called before the page is displayed and it created an object
89     * and display it in the navigation bar.
90 */
91 protected override void OnAppearing()
92 {
93     GetData data = new GetData();
94     data.SetLvl();
95
96     theLevel.Text = "LVL: " + GetData.lvl.ToString();
97 }
98 }
99 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.WorkPage"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models"
6     Title="Work Subcategories">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10        VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Work" TextColor="White" FontSize="20" FontAttributes="Italic"
12            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14            FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15            Margin="0,0,30,0"/>
16        </StackLayout>
17    </NavigationPage.TitleView>
18
19    <ContentPage.Content>
20        <ScrollView>
21            <Grid Margin="5,5,5,5">
22                <Grid.RowDefinitions>
23                    <RowDefinition Height="200" />
24                    <RowDefinition Height="200" />
25                    <RowDefinition Height="200" />
26                    <RowDefinition Height="200" />
27                    <RowDefinition Height="200" />
28                    <RowDefinition Height="200" />
29                    <RowDefinition Height="200" />
30                    <RowDefinition Height="200" />
31                </Grid.RowDefinitions>
32                <Grid.ColumnDefinitions>
33                    <ColumnDefinition />
34                </Grid.ColumnDefinitions>
35
36                <Image Grid.Column="0"
37                    Grid.Row="0"
38                    Aspect="AspectFill"
39                    Source="{local:ImageResource
40        Application_Green_Quake.Images.SubCategories.Travel.Carpool.jpg}"/>
41                <StackLayout Grid.Column="0"
42                    Grid.Row="0"
43                    BackgroundColor="Black"
44                    Opacity=".5">
45                    <StackLayout.GestureRecognizers>
46                        <TapGestureRecognizer Tapped="NavigateToCarpool"/>
47                    </StackLayout.GestureRecognizers>
48                </StackLayout>
49                <StackLayout Grid.Column="0"
50                    Grid.Row="0"
51                    VerticalOptions="End"
52                    Spacing="5"
53                    Margin="40,0,40,15">
54                    <StackLayout.GestureRecognizers>
55                        <TapGestureRecognizer Tapped="NavigateToCarpool"/>
56                    </StackLayout.GestureRecognizers>
57                    <Label Text="Carpool"
58                        TextColor="White"
59                        HorizontalOptions="Center"
60                        FontSize="24"
61                        FontAttributes="Bold"

```

```
58         FontFamily="Proxima Nova"/>
59         <Label Text="Carpool when traveling to work or school."
60             TextColor="White"
61             HorizontalOptions="Center"
62             FontSize="15"
63             FontFamily="Proxima Nova Thin"
64             HorizontalTextAlignment="Center"/>
65     </StackLayout>
66
67     <Image Grid.Column="0"
68         Grid.Row="1"
69         Aspect="AspectFill"
70         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.PublicTransport.jpg}"/>
71     <StackLayout Grid.Column="0"
72         Grid.Row="1"
73         BackgroundColor="Black"
74         Opacity=".5">
75         <StackLayout.GestureRecognizers>
76             <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
77         </StackLayout.GestureRecognizers>
78     </StackLayout>
79     <StackLayout Grid.Column="0"
80         Grid.Row="1"
81         VerticalOptions="End"
82         Spacing="5"
83         Margin="40,0,40,15">
84         <StackLayout.GestureRecognizers>
85             <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
86         </StackLayout.GestureRecognizers>
87         <Label Text="Public Transport"
88             TextColor="White"
89             HorizontalOptions="Center"
90             FontSize="24"
91             FontAttributes="Bold"
92             FontFamily="Proxima Nova"/>
93         <Label Text="Use Public Transport over a car."
94             TextColor="White"
95             HorizontalOptions="Center"
96             FontSize="15"
97             FontFamily="Proxima Nova Thin"
98             HorizontalTextAlignment="Center"/>
99     </StackLayout>
100
101     <Image Grid.Column="0"
102         Grid.Row="2"
103         Aspect="AspectFill"
104         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Cycle.jpg}"/>
105     <StackLayout Grid.Column="0"
106         Grid.Row="2"
107         BackgroundColor="Black"
108         Opacity=".5">
109         <StackLayout.GestureRecognizers>
110             <TapGestureRecognizer Tapped="NavigateToCycle"/>
111         </StackLayout.GestureRecognizers>
112     </StackLayout>
113     <StackLayout Grid.Column="0"
114         Grid.Row="2"
115         VerticalOptions="End"
116         Spacing="5"
```

```
117         Margin="40,0,40,15">
118         <StackLayout.GestureRecognizers>
119             <TapGestureRecognizer Tapped="NavigateToCycle"/>
120         </StackLayout.GestureRecognizers>
121         <Label Text="Cycle"
122             TextColor="White"
123             HorizontalOptions="Center"
124             FontSize="24"
125             FontAttributes="Bold"
126             FontFamily="Proxima Nova"/>
127         <Label Text="Cycle instead if you can."
128             TextColor="White"
129             HorizontalOptions="Center"
130             FontSize="15"
131             FontFamily="Proxima Nova Thin"
132             HorizontalTextAlignment="Center"/>
133     </StackLayout>
134
135     <Image Grid.Column="0"
136         Grid.Row="3"
137         Aspect="AspectFill"
138         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Walk.jpg}"/>
139     <StackLayout Grid.Column="0"
140         Grid.Row="3"
141         BackgroundColor="Black"
142         Opacity=".5">
143         <StackLayout.GestureRecognizers>
144             <TapGestureRecognizer Tapped="NavigateToWalk"/>
145         </StackLayout.GestureRecognizers>
146     </StackLayout>
147     <StackLayout Grid.Column="0"
148         Grid.Row="3"
149         VerticalOptions="End"
150         Spacing="5"
151         Margin="40,0,40,15">
152     <StackLayout.GestureRecognizers>
153         <TapGestureRecognizer Tapped="NavigateToWalk"/>
154     </StackLayout.GestureRecognizers>
155     <Label Text="Walk"
156         TextColor="White"
157         HorizontalOptions="Center"
158         FontSize="24"
159         FontAttributes="Bold"
160         FontFamily="Proxima Nova"/>
161     <Label Text="Walk there if you can."
162         TextColor="White"
163         HorizontalOptions="Center"
164         FontSize="15"
165         FontFamily="Proxima Nova Thin"
166         HorizontalTextAlignment="Center"/>
167 </StackLayout>
168
169     <Image Grid.Column="0"
170         Grid.Row="4"
171         Aspect="AspectFill"
172         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.EcoCar.jpg}"/>
173     <StackLayout Grid.Column="0"
174         Grid.Row="4"
175         BackgroundColor="Black"
176         Opacity=".5">
```

```

177         <StackLayout.GestureRecognizers>
178             <TapGestureRecognizer Tapped="NavigateToEcoFreindlyCar"/>
179         </StackLayout.GestureRecognizers>
180     </StackLayout>
181     <StackLayout Grid.Column="0"
182         Grid.Row="4"
183         VerticalOptions="End"
184         Spacing="5"
185         Margin="40,0,40,15">
186         <StackLayout.GestureRecognizers>
187             <TapGestureRecognizer Tapped="NavigateToEcoFreindlyCar"/>
188         </StackLayout.GestureRecognizers>
189         <Label Text="Eco Cart"
190             TextColor="White"
191             HorizontalOptions="Center"
192             FontSize="24"
193             FontAttributes="Bold"
194             FontFamily="Proxima Nova"/>
195         <Label Text="Purchase and use and Eco Friendly Car."
196             TextColor="White"
197             HorizontalOptions="Center"
198             FontSize="15"
199             FontFamily="Proxima Nova Thin"
200             HorizontalTextAlignment="Center"/>
201     </StackLayout>
202
203     <Image Grid.Column="0"
204         Grid.Row="5"
205         Aspect="AspectFill"
206         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Habits.OffLights.jpg}"/>
207     <StackLayout Grid.Column="0"
208         Grid.Row="5"
209         BackgroundColor="Black"
210         Opacity=".5">
211         <StackLayout.GestureRecognizers>
212             <TapGestureRecognizer Tapped="NavigateToOffLights"/>
213         </StackLayout.GestureRecognizers>
214     </StackLayout>
215     <StackLayout Grid.Column="0"
216         Grid.Row="5"
217         VerticalOptions="End"
218         Spacing="5"
219         Margin="40,0,40,15">
220         <StackLayout.GestureRecognizers>
221             <TapGestureRecognizer Tapped="NavigateToOffLights"/>
222         </StackLayout.GestureRecognizers>
223         <Label Text="Lights Off"
224             TextColor="White"
225             HorizontalOptions="Center"
226             FontSize="24"
227             FontAttributes="Bold"
228             FontFamily="Proxima Nova"/>
229         <Label Text="Turn off the lights when leaving the room."
230             TextColor="White"
231             HorizontalOptions="Center"
232             FontSize="15"
233             FontFamily="Proxima Nova Thin"
234             HorizontalTextAlignment="Center"/>
235     </StackLayout>
236

```

```
237         <Image Grid.Column="0"
238             Grid.Row="6"
239             Aspect="AspectFill"
240             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Work.Off.jpg}"/>
241         <StackLayout Grid.Column="0"
242             Grid.Row="6"
243             BackgroundColor="Black"
244             Opacity=".5">
245             <StackLayout.GestureRecognizers>
246                 <TapGestureRecognizer Tapped="NavigateToOffElectronics"/>
247             </StackLayout.GestureRecognizers>
248         </StackLayout>
249         <StackLayout Grid.Column="0"
250             Grid.Row="6"
251             VerticalOptions="End"
252             Spacing="5"
253             Margin="40,0,40,15">
254             <StackLayout.GestureRecognizers>
255                 <TapGestureRecognizer Tapped="NavigateToOffElectronics"/>
256             </StackLayout.GestureRecognizers>
257             <Label Text="Electronics Off"
258                 TextColor="White"
259                 HorizontalOptions="Center"
260                 FontSize="24"
261                 FontAttributes="Bold"
262                 FontFamily="Proxima Nova"/>
263             <Label Text="Switch off the Electronics that are not in use."
264                 TextColor="White"
265                 HorizontalOptions="Center"
266                 FontSize="15"
267                 FontFamily="Proxima Nova Thin"
268                 HorizontalTextAlignment="Center"/>
269         </StackLayout>
270
271         <Image Grid.Column="0"
272             Grid.Row="7"
273             Aspect="AspectFill"
274             Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Work.Remote.jpg}"/>
275         <StackLayout Grid.Column="0"
276             Grid.Row="7"
277             BackgroundColor="Black"
278             Opacity=".5">
279             <StackLayout.GestureRecognizers>
280                 <TapGestureRecognizer Tapped="NavigateToWorkingRemotely"/>
281             </StackLayout.GestureRecognizers>
282         </StackLayout>
283         <StackLayout Grid.Column="0"
284             Grid.Row="7"
285             VerticalOptions="End"
286             Spacing="5"
287             Margin="40,0,40,15">
288             <StackLayout.GestureRecognizers>
289                 <TapGestureRecognizer Tapped="NavigateToWorkingRemotely"/>
290             </StackLayout.GestureRecognizers>
291             <Label Text="Remote Work"
292                 TextColor="White"
293                 HorizontalOptions="Center"
294                 FontSize="24"
295                 FontAttributes="Bold"
296                 FontFamily="Proxima Nova"/>
```

```
297         <Label Text="Work Remotely if you can."
298             TextColor="White"
299             HorizontalOptions="Center"
300             FontSize="15"
301             FontFamily="Proxima Nova Thin"
302             HorizontalTextAlignment="Center"/>
303     </StackLayout>
304
305     <Image Grid.Column="0"
306         Grid.Row="8"
307         Aspect="AspectFill"
308         Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Work.Paper.jpg}"
309         Margin="0,0,0,5"/>
310     <StackLayout Grid.Column="0"
311         Grid.Row="8"
312         BackgroundColor="Black"
313         Opacity=".5"
314         Margin="0,0,0,5">
315         <StackLayout.GestureRecognizers>
316             <TapGestureRecognizer Tapped="NavigateToBothSidesPaper"/>
317         </StackLayout.GestureRecognizers>
318     </StackLayout>
319     <StackLayout Grid.Column="0"
320         Grid.Row="8"
321         VerticalOptions="End"
322         Spacing="5"
323         Margin="40,0,40,15">
324         <StackLayout.GestureRecognizers>
325             <TapGestureRecognizer Tapped="NavigateToBothSidesPaper"/>
326         </StackLayout.GestureRecognizers>
327         <Label Text="Both Sides"
328             TextColor="White"
329             HorizontalOptions="Center"
330             FontSize="24"
331             FontAttributes="Bold"
332             FontFamily="Proxima Nova"/>
333         <Label Text="Use Both Sides of the paper when you are using it."
334             TextColor="White"
335             HorizontalOptions="Center"
336             FontSize="15"
337             FontFamily="Proxima Nova Thin"
338             HorizontalTextAlignment="Center"/>
339     </StackLayout>
340 </Grid>
341 </ScrollView>
342 </ContentPage.Content>
343 </ContentPage>
```



```
1 /!* \class The WorkPage View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the WorkPage View Class. This page displays and allows the
   navigation to each of the actions in the Work category.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Application_Green_Quake.Views.EcoActions.Habits;
11 using Application_Green_Quake.Views.EcoActions.Travel;
12 using Application_Green_Quake.Views.EcoActions.Work;
13 using System;
14 using Xamarin.Forms;
15 using Xamarin.Forms.Xaml;
16
17 namespace Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu
18 {
19     [XamlCompilation(XamlCompilationOptions.Compile)]
20     public partial class WorkPage : ContentPage
21     {
22         public WorkPage()
23         {
24             InitializeComponent();
25             OnAppearing();
26         }
27         /** This function navigates to Carpool.
28         */
29         private async void NavigateToCarpool(object sender, EventArgs e)
30         {
31             await Navigation.PushAsync(new Carpool());
32         }
33         /** This function navigates to PublicTransport.
34         */
35         private async void NavigateToPublicTransport(object sender, EventArgs e)
36         {
37             await Navigation.PushAsync(new PublicTransport());
38         }
39         /** This function navigates to Cycle.
40         */
41         private async void NavigateToCycle(object sender, EventArgs e)
42         {
43             await Navigation.PushAsync(new Cycle());
44         }
45         /** This function navigates to Walk.
46         */
47         private async void NavigateToWalk(object sender, EventArgs e)
48         {
49             await Navigation.PushAsync(new Walk());
50         }
51         /** This function navigates to EcoFreindlyCar.
52         */
53         private async void NavigateToEcoFreindlyCar(object sender, EventArgs e)
54         {
55             await Navigation.PushAsync(new EcoFreindlyCar());
56         }
57         /** This function navigates to TurnOffLights.
58         */
59         private async void NavigateToOffLights(object sender, EventArgs e)
```



```
60     {
61         await Navigation.PushAsync(new TurnOffLights());
62     }
63     /** This function navigates to OffElectronics.
64     */
65     private async void NavigateToOffElectronics(object sender, EventArgs e)
66     {
67         await Navigation.PushAsync(new OffElectronics());
68     }
69     /** This function navigates to WorkingRemotely.
70     */
71     private async void NavigateToWorkingRemotely(object sender, EventArgs e)
72     {
73         await Navigation.PushAsync(new WorkingRemotely());
74     }
75     /** This function navigates to BothSidesPaper.
76     */
77     private async void NavigateToBothSidesPaper(object sender, EventArgs e)
78     {
79         await Navigation.PushAsync(new BothSidesPaper());
80     }
81     /** This function is called before the page is displayed and it created an object
82     ans uses its SetLvl method to set the players level in the app
83     * and display it in the navigation bar.
84     */
85     protected override void OnAppearing()
86     {
87         GetData data = new GetData();
88         data.SetLvl();
89
90         theLevel.Text = "LVL: " + GetData.lvl.ToString();
91     }
92 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.ProfilePage.Achievements"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6     <ContentPage.Content>
7         <ScrollView>
8             <Grid BackgroundColor="#002a1e" RowSpacing="0" Padding="10,10,10,10">
9                 <Grid.ColumnDefinitions>
10                    <ColumnDefinition Width="*"/>
11                    <ColumnDefinition Width="*"/>
12                    <ColumnDefinition Width="*"/>
13                    <ColumnDefinition Width="*"/>
14                </Grid.ColumnDefinitions>
15                <Grid.RowDefinitions>
16                    <RowDefinition Height="3200"/>
17                </Grid.RowDefinitions>
18
19                <StackLayout Grid.Column="0" Grid.Row="0" VerticalOptions="Start">
20                    <Frame CornerRadius="60" HorizontalOptions="Start" Margin="0"
21                    Padding="0" BackgroundColor="#33554b" >
22                        <Image x:Name="a1" Source="{local:ImageResource
23                    Application_Green_Quake.Images.lockTwo.png}" Aspect="AspectFill"/>
24                    </Frame>
25
26                    <Label x:Name="a1Txt" Text="Locked " TextColor="White"
27                    HorizontalTextAlignment="Center" HeightRequest="60"/>
28
29                    <Frame CornerRadius="60" HorizontalOptions="Start"
30                    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
31                        <Image x:Name="a5" Source="{local:ImageResource
32                    Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
33                    </Frame>
34
35                    <Label x:Name="a5Txt" Text="Locked " TextColor="White"
36                    HorizontalTextAlignment="Center" HeightRequest="60"/>
37
38                    <Frame CornerRadius="60" HorizontalOptions="Start"
39                    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
40                        <Image x:Name="a9" Source="{local:ImageResource
41                    Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
42                    </Frame>
43
44                    <Label x:Name="a9Txt" Text="Locked " TextColor="White"
45                    HorizontalTextAlignment="Center" HeightRequest="60"/>
46
47                    <Frame CornerRadius="60" HorizontalOptions="Start"
48                    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
49                        <Image x:Name="a13" Source="{local:ImageResource
50                    Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
51                    </Frame>
52
53                    <Label x:Name="a13Txt" Text="Locked " TextColor="White"
54                    HorizontalTextAlignment="Center" HeightRequest="60"/>
55
56                    <Frame CornerRadius="60" HorizontalOptions="Start"
57                    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
58                        <Image x:Name="a17" Source="{local:ImageResource
59                    Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
60                    </Frame>
61
62                    <Label x:Name="a17Txt" Text="Locked " TextColor="White"
63                    HorizontalTextAlignment="Center" HeightRequest="60"/>
64
65                    <Frame CornerRadius="60" HorizontalOptions="Start"
66                    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
67                        <Image x:Name="a21" Source="{local:ImageResource
68                    Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
69                    </Frame>
70
71                    <Label x:Name="a21Txt" Text="Locked " TextColor="White"
72                    HorizontalTextAlignment="Center" HeightRequest="60"/>
73                </StackLayout>
74            </Grid>
75        </ScrollView>
76    </ContentPage.Content>
77 </ContentPage>
```

```
48         </Frame>
49
50         <Label x:Name="a21Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
51         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
52             <Image x:Name="a25" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
53         </Frame>
54
55         <Label x:Name="a25Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
56
57         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
58             <Image x:Name="a29" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
59         </Frame>
60
61         <Label x:Name="a29Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
62         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
63             <Image x:Name="a33" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
64         </Frame>
65
66         <Label x:Name="a33Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
67         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
68             <Image x:Name="a37" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
69         </Frame>
70
71         <Label x:Name="a37Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
72         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
73             <Image x:Name="a41" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
74         </Frame>
75
76         <Label x:Name="a41Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
77         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
78             <Image x:Name="a45" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
79         </Frame>
80
81         <Label x:Name="a45Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
82         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
83             <Image x:Name="a49" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
84         </Frame>
85
86         <Label x:Name="a49Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
87         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
88             <Image x:Name="a53" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
```

```
89         </Frame>
90
91         <Label x:Name="a53Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
92         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
93             <Image x:Name="a57" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
94         </Frame>
95
96         <Label x:Name="a57Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
97         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
98             <Image x:Name="a61" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
99         </Frame>
100
101        <Label x:Name="a61Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
102        <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
103            <Image x:Name="a65" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
104        </Frame>
105
106        <Label x:Name="a65Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
107        <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
108            <Image x:Name="a69" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
109        </Frame>
110
111        <Label x:Name="a69Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
112        <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
113            <Image x:Name="a73" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
114        </Frame>
115
116        <Label x:Name="a73Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
117        <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
118            <Image x:Name="a77" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
119        </Frame>
120
121        <Label x:Name="a77Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
122        <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
123            <Image x:Name="a81" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
124        </Frame>
125
126        <Label x:Name="a81Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
127    </StackLayout>
128
129    <StackLayout Grid.Column="1" Grid.Row="0" VerticalOptions="Start">
130        <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b" >
```

```
131         <Image x:Name="a2" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
132     </Frame>
133
134     <Label x:Name="a2Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
135     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
136         <Image x:Name="a6" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
137     </Frame>
138
139     <Label x:Name="a6Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
140     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
141         <Image x:Name="a10" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
142     </Frame>
143
144     <Label x:Name="a10Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
145     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b" >
146         <Image x:Name="a14" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
147     </Frame>
148
149     <Label x:Name="a14Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
150     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
151         <Image x:Name="a18" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
152     </Frame>
153
154     <Label x:Name="a18Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
155     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
156         <Image x:Name="a22" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
157     </Frame>
158
159     <Label x:Name="a22Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
160     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
161         <Image x:Name="a26" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
162     </Frame>
163
164     <Label x:Name="a26Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
165     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
166         <Image x:Name="a30" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
167     </Frame>
168
169     <Label x:Name="a30Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
170     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
171         <Image x:Name="a34" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
```



```
172         </Frame>
173
174         <Label x:Name="a34Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
175         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
176         <Image x:Name="a38" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
177         </Frame>
178
179         <Label x:Name="a38Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
180         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
181         <Image x:Name="a42" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
182         </Frame>
183
184         <Label x:Name="a42Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
185         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
186         <Image x:Name="a46" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
187         </Frame>
188
189         <Label x:Name="a46Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
190         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
191         <Image x:Name="a50" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
192         </Frame>
193
194         <Label x:Name="a50Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
195         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
196         <Image x:Name="a54" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
197         </Frame>
198
199         <Label x:Name="a54Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
200         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
201         <Image x:Name="a58" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
202         </Frame>
203
204         <Label x:Name="a58Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
205         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
206         <Image x:Name="a62" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
207         </Frame>
208
209         <Label x:Name="a62Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
210         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
211         <Image x:Name="a66" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
212         </Frame>
```

```
213
214         <Label x:Name="a66Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
215         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
216             <Image x:Name="a70" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
217         </Frame>
218
219         <Label x:Name="a70Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
220         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
221             <Image x:Name="a74" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
222         </Frame>
223
224         <Label x:Name="a74Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
225         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
226             <Image x:Name="a78" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
227         </Frame>
228
229         <Label x:Name="a78Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
230     </StackLayout>
231
232     <StackLayout Grid.Column="2" Grid.Row="0" VerticalOptions="Start">
233         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
234             <Image x:Name="a3" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
235         </Frame>
236
237         <Label x:Name="a3Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
238         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
239             <Image x:Name="a7" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
240         </Frame>
241
242         <Label x:Name="a7Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
243         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
244             <Image x:Name="a11" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
245         </Frame>
246
247         <Label x:Name="a11Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
248         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
249             <Image x:Name="a15" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
250         </Frame>
251
252         <Label x:Name="a15Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
253         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
254             <Image x:Name="a19" Source="{local:ImageResource
```

```
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
255     </Frame>
256
257     <Label x:Name="a19Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
258     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
259         <Image x:Name="a23" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
260     </Frame>
261
262     <Label x:Name="a23Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
263     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
264         <Image x:Name="a27" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
265     </Frame>
266
267     <Label x:Name="a27Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
268     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
269         <Image x:Name="a31" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
270     </Frame>
271
272     <Label x:Name="a31Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
273     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
274         <Image x:Name="a35" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
275     </Frame>
276
277     <Label x:Name="a35Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
278     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
279         <Image x:Name="a39" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
280     </Frame>
281
282     <Label x:Name="a39Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
283     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
284         <Image x:Name="a43" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
285     </Frame>
286
287     <Label x:Name="a43Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
288     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
289         <Image x:Name="a47" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
290     </Frame>
291
292     <Label x:Name="a47Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
293     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
294         <Image x:Name="a51" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
```



```
295         </Frame>
296
297         <Label x:Name="a51Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
298         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
299             <Image x:Name="a55" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
300         </Frame>
301
302         <Label x:Name="a55Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
303         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
304             <Image x:Name="a59" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
305         </Frame>
306
307         <Label x:Name="a59Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
308         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
309             <Image x:Name="a63" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
310         </Frame>
311
312         <Label x:Name="a63Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
313         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
314             <Image x:Name="a67" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
315         </Frame>
316
317         <Label x:Name="a67Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
318         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
319             <Image x:Name="a71" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
320         </Frame>
321
322         <Label x:Name="a71Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
323         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
324             <Image x:Name="a75" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
325         </Frame>
326
327         <Label x:Name="a75Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
328         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
329             <Image x:Name="a79" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
330         </Frame>
331
332         <Label x:Name="a79Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
333     </StackLayout>
334
335     <StackLayout Grid.Column="3" Grid.Row="0" VerticalOptions="Start">
336         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
```

```
337         <Image x:Name="a4" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
338     </Frame>
339
340     <Label x:Name="a4Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
341     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
342         <Image x:Name="a8" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
343     </Frame>
344
345     <Label x:Name="a8Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
346     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
347         <Image x:Name="a12" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
348     </Frame>
349
350     <Label x:Name="a12Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
351     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
352         <Image x:Name="a16" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
353     </Frame>
354
355     <Label x:Name="a16Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
356     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
357         <Image x:Name="a20" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
358     </Frame>
359
360     <Label x:Name="a20Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
361     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
362         <Image x:Name="a24" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
363     </Frame>
364
365     <Label x:Name="a24Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
366     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
367         <Image x:Name="a28" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
368     </Frame>
369
370     <Label x:Name="a28Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
371     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
372         <Image x:Name="a32" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
373     </Frame>
374
375     <Label x:Name="a32Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
376     <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
377         <Image x:Name="a36" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
```

```
378         </Frame>
379
380         <Label x:Name="a36Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
381         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
382         <Image x:Name="a40" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
383         </Frame>
384
385         <Label x:Name="a40Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
386         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
387         <Image x:Name="a44" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
388         </Frame>
389
390         <Label x:Name="a44Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
391         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
392         <Image x:Name="a48" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
393         </Frame>
394
395         <Label x:Name="a48Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
396         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
397         <Image x:Name="a52" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
398         </Frame>
399
400         <Label x:Name="a52Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
401         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
402         <Image x:Name="a56" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
403         </Frame>
404
405         <Label x:Name="a56Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
406         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
407         <Image x:Name="a60" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
408         </Frame>
409
410         <Label x:Name="a60Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
411         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
412         <Image x:Name="a64" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
413         </Frame>
414
415         <Label x:Name="a64Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
416         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
417         <Image x:Name="a68" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
418         </Frame>
```

```
419
420         <Label x:Name="a68Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
421         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
422             <Image x:Name="a72" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
423         </Frame>
424
425         <Label x:Name="a72Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
426         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
427             <Image x:Name="a76" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
428         </Frame>
429
430         <Label x:Name="a76Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
431         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
432             <Image x:Name="a80" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
433         </Frame>
434
435         <Label x:Name="a80Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center" HeightRequest="60"/>
436     </StackLayout>
437 </Grid>
438 </ScrollView>
439 </ContentPage.Content>
440 </ContentPage>
```

```

1  /!* \class The Achievements View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the Achievements View Class. This class is the class that displays a
   Achievements on the Achievements page.
7  *
8  */
9  using Application_Green_Quake.ViewModels;
10 using Xamarin.Forms;
11 using Xamarin.Forms.Xaml;
12
13 namespace Application_Green_Quake.Views.ProfilePage
14 {
15     [XamlCompilation(XamlCompilationOptions.Compile)]
16     public partial class Achievements : ContentPage
17     {
18         IAuth auth;
19
20         public Achievements()
21         {
22             InitializeComponent();
23             auth = DependencyService.Get<IAuth>();
24             OnAppearing();
25         }
26         /** This function is called before the page is displayed. It displays the images as
are met
27         */
28         protected override void OnAppearing()
29         {
30             if (GetAchievementsData.brushingCount >= 5 && GetAchievementsData.brushingCount
31             {
32                 a1.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingBronze.
33                 a1Txt.Text = "Brushing Bronze";
34             }
35             else if (GetAchievementsData.brushingCount >= 15 && GetAchievementsData.brushing
36             {
37                 a1.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingSilver.
38                 a1Txt.Text = "Brushing Silver";
39             }
40             else if (GetAchievementsData.brushingCount >= 25)
41             {
42                 a1.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingGold.pn
43                 a1Txt.Text = "Brushing Gold";
44             }
45
46             if (GetAchievementsData.fullWasherCount >= 5 && GetAchievementsData.fullWasherCo
47             {
48                 a2.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterBronz
49                 a2Txt.Text = "Dish Washer Bronze";
50             }
51             else if (GetAchievementsData.fullWasherCount >= 15 && GetAchievementsData.fullWa
52             {
53                 a2.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterSilve
54                 a2Txt.Text = "Dish Washer Silver";

```

```
55     }
56     else if (GetAchievementsData.fullWasherCount >= 25)
57     {
58         a2.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterGold.
59         a2Txt.Text = "Dish Washer Gold";
60     }
61
62     if (GetAchievementsData.shareCount >= 5 && GetAchievementsData.shareCount < 15)
63     {
64         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerBronze.pn
65         a3Txt.Text = "Shower Bronze";
66     }
67     else if (GetAchievementsData.shareCount >= 15 && GetAchievementsData.shareCount
68     {
69         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerSilver.pn
70         a3Txt.Text = "Shower Silver";
71     }
72     else if (GetAchievementsData.shareCount >= 25)
73     {
74         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerGold.png"
75         a3Txt.Text = "Shower Gold";
76     }
77
78     if (GetAchievementsData.timedShowerCount >= 5 && GetAchievementsData.timedShower
79     {
80         a4.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerBronz
81         a4Txt.Text = "Timed Shower Bronze";
82     }
83     else if (GetAchievementsData.timedShowerCount >= 15 && GetAchievementsData.timed
25)
84     {
85         a4.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerSilve
86         a4Txt.Text = "Timed Shower Silver";
87     }
88     else if (GetAchievementsData.timedShowerCount >= 25)
89     {
90         a4.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerGold.
91         a4Txt.Text = "Timed Shower Gold";
92     }
93
94     if (GetAchievementsData.offLigtsCount >= 5 && GetAchievementsData.offLigtsCount
95     {
96         a5.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightBronze.png
97         a5Txt.Text = "Off Lights Bronze";
98     }
99     else if (GetAchievementsData.offLigtsCount >= 15 && GetAchievementsData.offLigts
100     {
101         a5.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightSilver.png
102         a5Txt.Text = "Off Lights Silver";
103     }
104     else if (GetAchievementsData.offLigtsCount >= 25)
105     {
106         a5.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightGold.png")
```



```
107         a5Txt.Text = "Off Lights Gold";
108     }
109
110     if (GetAchievementsData.matchesCount >= 5 && GetAchievementsData.matchesCount <
111     {
112         a6.Source =
113         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.matchBronze.png");
114         a6Txt.Text = "Matches Bronze";
115     }
116     else if (GetAchievementsData.matchesCount >= 15 && GetAchievementsData.matchesCo
117     {
118         a6.Source =
119         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.matchSilver.png");
120         a6Txt.Text = "Matches Silver";
121     }
122     else if (GetAchievementsData.matchesCount >= 25)
123     {
124         a6.Source =
125         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.matchGold.png");
126         a6Txt.Text = "Matches Gold";
127     }
128
129     if (GetAchievementsData.foodDeliverCount >= 5 && GetAchievementsData.foodDeliver
130     {
131         a7.Source =
132         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.deliverBronze.png");
133         a7Txt.Text = "Food Delivered Bronze";
134     }
135     else if (GetAchievementsData.foodDeliverCount >= 15 && GetAchievementsData.foodD
136     25)
137     {
138         a7.Source =
139         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.deliverSilver.png");
140         a7Txt.Text = "Food Delivered Silver";
141     }
142     else if (GetAchievementsData.foodDeliverCount >= 25)
143     {
144         a7.Source =
145         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.deliverGold.png");
146         a7Txt.Text = "Food Delivered Gold";
147     }
148
149     if (GetAchievementsData.eatAllCount >= 5 && GetAchievementsData.eatAllCount < 15
150     {
151         a8.Source =
152         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.eatAllBronze.png");
153         a8Txt.Text = "Eat All Bronze";
154     }
155     else if (GetAchievementsData.eatAllCount >= 15 && GetAchievementsData.eatAllCoun
156     {
157         a8.Source =
158         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.eatAllSilver.png");
159         a8Txt.Text = "Eat All Silver";
160     }
161     else if (GetAchievementsData.eatAllCount >= 25)
162     {
163         a8.Source =
164         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.eatAllGold.png");
165         a8Txt.Text = "Eat All Gold";
166     }
167
168     if (GetAchievementsData.saveLeftOversCount >= 5 && GetAchievementsData.saveLeftO
169     {
```

```
160         a9.Source =
161     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.leftoversBronze.png
162         a9Txt.Text = "Save Leftovers Bronze";
163     }
164     else if (GetAchievementsData.saveLeftOversCount >= 15 && GetAchievementsData.sav
165     < 25)
166     {
167         a9.Source =
168     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.leftoversSilver.png
169         a9Txt.Text = "Save Leftovers Silver";
170     }
171     else if (GetAchievementsData.saveLeftOversCount >= 25)
172     {
173         a9.Source =
174     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.leftoversGold.png")
175         a9Txt.Text = "Save Leftovers Gold";
176     }
177     if (GetAchievementsData.noMeatCount >= 5 && GetAchievementsData.noMeatCount < 15
178     {
179         a10.Source =
180     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.noMeatBronze.png");
181         a10Txt.Text = "No Meat Bronze";
182     }
183     else if (GetAchievementsData.noMeatCount >= 15 && GetAchievementsData.noMeatCoun
184     {
185         a10.Source =
186     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.noMeatSilver.png");
187         a10Txt.Text = "No Meat Silver";
188     }
189     else if (GetAchievementsData.noMeatCount >= 25)
190     {
191         a10.Source =
192     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.noMeatGold.png");
193         a10Txt.Text = "No Meat Gold";
194     }
195     if (GetAchievementsData.organicFoodCount >= 5 && GetAchievementsData.organicFood
196     {
197         a11.Source =
198     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.organicBronze.png")
199         a11Txt.Text = "Organic Food Bronze";
200     }
201     else if (GetAchievementsData.organicFoodCount >= 15 && GetAchievementsData.organ
202     25)
203     {
204         a11.Source =
205     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.organicSilver.png")
206         a11Txt.Text = "Organic Food Silver";
207     }
208     else if (GetAchievementsData.organicFoodCount >= 25)
209     {
210         a11.Source =
211     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.organicGold.png");
212         a11Txt.Text = "Organic Food Gold";
213     }
214     if (GetAchievementsData.ownCoffeeCount >= 5 && GetAchievementsData.ownCoffeeCoun
215     {
216         a12.Source =
217     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.ownCoffeeBronze.png
218         a12Txt.Text = "Own Coffee Bronze";
219     }
220     }
```



```
211         else if (GetAchievementsData.ownCoffeeCount >= 15 && GetAchievementsData.ownCoff
212         {
213             a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.ownCoffeeSilver.png
214             a12Txt.Text = "Own Coffee Silver";
215         }
216         else if (GetAchievementsData.ownCoffeeCount >= 25)
217         {
218             a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.ownCoffeeGold.png")
219             a12Txt.Text = "Own Coffee Gold";
220         }
221
222         if (GetAchievementsData.reCoffeeMugCount >= 5 && GetAchievementsData.reCoffeeMug
223         {
224             a13.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.reCupBronze.png");
225             a13Txt.Text = "Reusable Mug Bronze";
226         }
227         else if (GetAchievementsData.reCoffeeMugCount >= 15 && GetAchievementsData.reCof
228         25)
229         {
230             a13.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.reCupSilver.png");
231             a13Txt.Text = "Reusable Mug Silver";
232         }
233         else if (GetAchievementsData.reCoffeeMugCount >= 25)
234         {
235             a13.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.reCupGold.png");
236             a13Txt.Text = "Reusable Mug Gold";
237         }
238         if (GetAchievementsData.steelStrawCount >= 5 && GetAchievementsData.steelStrawCo
239         {
240             a14.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawBronze.png");
241             a14Txt.Text = "Steel Straw Bronze";
242         }
243         else if (GetAchievementsData.steelStrawCount >= 15 && GetAchievementsData.steelS
244         {
245             a14.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawSilver.png");
246             a14Txt.Text = "Steel Straw Silver";
247         }
248         else if (GetAchievementsData.steelStrawCount >= 25)
249         {
250             a14.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawGold.png");
251             a14Txt.Text = "Steel Straw Gold";
252         }
253
254         if (GetAchievementsData.waterOverFizzyCount >= 5 && GetAchievementsData.waterOve
255         15)
256         {
257             a15.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterBronze.png");
258             a15Txt.Text = "Water Bronze";
259         }
260         else if (GetAchievementsData.waterOverFizzyCount >= 15 &&
GetAchievementsData.waterOverFizzyCount < 25)
261         {
262             a15.Source =
```

```
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterSilver.png");
262     a15Txt.Text = "Water Silver";
263     }
264     else if (GetAchievementsData.waterOverFizzyCount >= 25)
265     {
266         a15.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterGold.png");
267         a15Txt.Text = "Water Gold";
268     }
269
270     if (GetAchievementsData.fullDryerCount >= 5 && GetAchievementsData.fullDryerCoun
271     {
272         a16.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerBronze.png
273         a16Txt.Text = "Dryer Bronze";
274     }
275     else if (GetAchievementsData.fullDryerCount >= 15 && GetAchievementsData.fullDry
276     {
277         a16.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerSilver.png
278         a16Txt.Text = "Dryer Silver";
279     }
280     else if (GetAchievementsData.fullDryerCount >= 25)
281     {
282         a16.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerGold.png")
283         a16Txt.Text = "Dryer Gold";
284     }
285
286     if (GetAchievementsData.hangDryCount >= 5 && GetAchievementsData.hangDryCount <
287     {
288         a17.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangBronze.png"
289         a17Txt.Text = "Hang Dry Bronze";
290     }
291     else if (GetAchievementsData.hangDryCount >= 15 && GetAchievementsData.hangDryCo
292     {
293         a17.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangSilver.png"
294         a17Txt.Text = "Hang Dry Silver";
295     }
296     else if (GetAchievementsData.hangDryCount >= 25)
297     {
298         a17.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangGold.png");
299         a17Txt.Text = "Hang Dry Gold";
300     }
301
302     if (GetAchievementsData.microwaveCount >= 5 && GetAchievementsData.microwaveCoun
303     {
304         a18.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveBronze
305         a18Txt.Text = "Microwave Bronze";
306     }
307     else if (GetAchievementsData.microwaveCount >= 15 && GetAchievementsData.microwa
308     {
309         a18.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveSilver
310         a18Txt.Text = "Microwave Silver";
311     }
312     else if (GetAchievementsData.microwaveCount >= 25)
313     {
314         a18.Source =
```

```
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveGold.p
315     a18Txt.Text = "Microwave Gold";
316     }
317
318     if (GetAchievementsData.reBatteriesCount >= 5 && GetAchievementsData.reBatteries
319     {
320     a19.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesBron
321     a19Txt.Text = "Reusable Batteries Bronze";
322     }
323     else if (GetAchievementsData.reBatteriesCount >= 15 && GetAchievementsData.reBat
25)
324     {
325     a19.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesSilv
326     a19Txt.Text = "Reusable Batteries Silver";
327     }
328     else if (GetAchievementsData.reBatteriesCount >= 25)
329     {
330     a19.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesGold
331     a19Txt.Text = "Reusable Batteries Gold";
332     }
333
334     if (GetAchievementsData.offSocketCount >= 5 && GetAchievementsData.offSocketCoun
335     {
336     a20.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.socketBronze.pn
337     a20Txt.Text = "Socket Off Bronze";
338     }
339     else if (GetAchievementsData.offSocketCount >= 15 && GetAchievementsData.offSock
340     {
341     a20.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.socketSilver.pn
342     a20Txt.Text = "Socket Off Silver";
343     }
344     else if (GetAchievementsData.offSocketCount >= 25)
345     {
346     a20.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.socketGold.png"
347     a20Txt.Text = "Socket Off Gold";
348     }
349
350     if (GetAchievementsData.fullWasherCount >= 5 && GetAchievementsData.fullWasherCo
351     {
352     a21.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineB
353     a21Txt.Text = "Washing Machine Bronze";
354     }
355     else if (GetAchievementsData.fullWasherCount >= 15 && GetAchievementsData.fullWa
356     {
357     a21.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineS
358     a21Txt.Text = "Washing Machine Silver";
359     }
360     else if (GetAchievementsData.fullWasherCount >= 25)
361     {
362     a21.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineG
363     a21Txt.Text = "Washing Machine Gold";
364     }
365
366     if (GetAchievementsData.carpoolCount >= 5 && GetAchievementsData.carpoolCount <
```

```
367     {
368         a22.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolBronze.p
369         a22Txt.Text = "Carpool Bronze";
370     }
371     else if (GetAchievementsData.carpoolCount >= 15 && GetAchievementsData.carpoolCo
372     {
373         a22.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolSilver.p
374         a22Txt.Text = "Carpool Silver";
375     }
376     else if (GetAchievementsData.carpoolCount >= 25)
377     {
378         a22.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolGold.png
379         a22Txt.Text = "Carpool Gold";
380     }
381
382     if (GetAchievementsData.cycleCount >= 5 && GetAchievementsData.cycleCount < 15)
383     {
384         a23.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleBronze.png
385         a23Txt.Text = "Cycle Bronze";
386     }
387     else if (GetAchievementsData.cycleCount >= 15 && GetAchievementsData.cycleCount
388     {
389         a23.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleSilver.png
390         a23Txt.Text = "Cycle Silver";
391     }
392     else if (GetAchievementsData.cycleCount >= 25)
393     {
394         a23.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleGold.png")
395         a23Txt.Text = "Cycle Gold";
396     }
397
398     if (GetAchievementsData.ecoCarCount >= 5 && GetAchievementsData.ecoCarCount < 15
399     {
400         a24.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarBronze.png"
401         a24Txt.Text = "Eco Car Bronze";
402     }
403     else if (GetAchievementsData.ecoCarCount >= 15 && GetAchievementsData.ecoCarCoun
404     {
405         a24.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarSilver.png"
406         a24Txt.Text = "Eco Car Silver";
407     }
408     else if (GetAchievementsData.ecoCarCount >= 25)
409     {
410         a24.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarGold.png");
411         a24Txt.Text = "Eco Car Gold";
412     }
413
414     if (GetAchievementsData.transportCount >= 5 && GetAchievementsData.transportCoun
415     {
416         a25.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbBronze.png");
417         a25Txt.Text = "Public Transport Bronze";
418     }
419     else if (GetAchievementsData.transportCount >= 15 && GetAchievementsData.transpo
```

```
420     {
421         a25.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbSilver.png");
422         a25Txt.Text = "Public Transport Silver";
423     }
424     else if (GetAchievementsData.transportCount >= 25)
425     {
426         a25.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbGold.png");
427         a25Txt.Text = "Public Transport Gold";
428     }
429
430     if (GetAchievementsData.walkCount >= 5 && GetAchievementsData.walkCount < 15)
431     {
432         a26.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkBronze.png"
433         a26Txt.Text = "Walk Bronze";
434     }
435     else if (GetAchievementsData.walkCount >= 15 && GetAchievementsData.walkCount <
436     {
437         a26.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkSilver.png"
438         a26Txt.Text = "Walk Silver";
439     }
440     else if (GetAchievementsData.walkCount >= 25)
441     {
442         a26.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkGold.png");
443         a26Txt.Text = "Walk Gold";
444     }
445
446     if (GetAchievementsData.applianceCount >= 5 && GetAchievementsData.applianceCoun
447     {
448         a27.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceBron
449         a27Txt.Text = "Eco Appliance Bronze";
450     }
451     else if (GetAchievementsData.applianceCount >= 15 && GetAchievementsData.applian
452     {
453         a27.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceSilv
454         a27Txt.Text = "Eco Appliance Silver";
455     }
456     else if (GetAchievementsData.applianceCount >= 25)
457     {
458         a27.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceGold
459         a27Txt.Text = "Eco Appliance Gold";
460     }
461
462     if (GetAchievementsData.foodCount >= 5 && GetAchievementsData.foodCount < 15)
463     {
464         a28.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkBronze.pn
465         a28Txt.Text = "Food In Bulk Bronze";
466     }
467     else if (GetAchievementsData.foodCount >= 15 && GetAchievementsData.foodCount <
468     {
469         a28.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkSilver.pn
470         a28Txt.Text = "Food In Bulk Silver";
471     }
472     else if (GetAchievementsData.foodCount >= 25)
```

```

473     {
474         a28.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkGold.png"
475         a28Txt.Text = "Food In Bulk Gold";
476     }
477
478     if (GetAchievementsData.clothesCount >= 5 && GetAchievementsData.clothesCount <
479     {
480         a29.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalBronze
481         a29Txt.Text = "Ethical Clothing Bronze";
482     }
483     else if (GetAchievementsData.clothesCount >= 15 && GetAchievementsData.clothesCo
484     {
485         a29.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalSilver
486         a29Txt.Text = "Ethical Clothing Silver";
487     }
488     else if (GetAchievementsData.clothesCount >= 25)
489     {
490         a29.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalGold.p
491         a29Txt.Text = "Ethical Clothing Gold";
492     }
493
494     if (GetAchievementsData.localCount >= 5 && GetAchievementsData.localCount < 15)
495     {
496         a30.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localBronze.p
497         a30Txt.Text = "Buy Local Bronze";
498     }
499     else if (GetAchievementsData.localCount >= 15 && GetAchievementsData.localCount
500     {
501         a30.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localSilver.p
502         a30Txt.Text = "Buy Local Silver";
503     }
504     else if (GetAchievementsData.localCount >= 25)
505     {
506         a30.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localGold.png
507         a30Txt.Text = "Buy Local Gold";
508     }
509
510     if (GetAchievementsData.clothNapkinCount >= 5 && GetAchievementsData.clothNapkin
511     {
512         a31.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinBronze.
513         a31Txt.Text = "Napkin Bronze";
514     }
515     else if (GetAchievementsData.clothNapkinCount >= 15 && GetAchievementsData.cloth
25)
516     {
517         a31.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinSilver.
518         a31Txt.Text = "Napkin Silver";
519     }
520     else if (GetAchievementsData.clothNapkinCount >= 25)
521     {
522         a31.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinGold.pn
523         a31Txt.Text = "Napkin Gold";
524     }

```



```
525
526     if (GetAchievementsData.productCount >= 5 && GetAchievementsData.productCount <
527     {
528         a32.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productBronze
529         a32Txt.Text = "Eco Product Bronze";
530     }
531     else if (GetAchievementsData.productCount >= 15 && GetAchievementsData.productCo
532     {
533         a32.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productSilver
534         a32Txt.Text = "Eco Product Silver";
535     }
536     else if (GetAchievementsData.productCount >= 25)
537     {
538         a32.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productGold.p
539         a32Txt.Text = "Eco Product Gold";
540     }
541
542     if (GetAchievementsData.reBagCount >= 5 && GetAchievementsData.reBagCount < 15)
543     {
544         a33.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagBronze.p
545         a33Txt.Text = "Reusable Bag Bronze";
546     }
547     else if (GetAchievementsData.reBagCount >= 15 && GetAchievementsData.reBagCount
548     {
549         a33.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagSilver.p
550         a33Txt.Text = "Reusable Bag Silver";
551     }
552     else if (GetAchievementsData.reBagCount >= 25)
553     {
554         a33.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagGold.png
555         a33Txt.Text = "Reusable Bag Gold";
556     }
557
558     if (GetAchievementsData.looseLeafCount >= 5 && GetAchievementsData.looseLeafCoun
559     {
560         a34.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaBronze.png
561         a34Txt.Text = "Loose Leaf Bronze";
562     }
563     else if (GetAchievementsData.looseLeafCount >= 15 && GetAchievementsData.looseLe
564     {
565         a34.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaSilver.png
566         a34Txt.Text = "Loose Leaf Silver";
567     }
568     else if (GetAchievementsData.looseLeafCount >= 25)
569     {
570         a34.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaGold.png")
571         a34Txt.Text = "Loose Leaf Gold";
572     }
573
574     if (GetAchievementsData.toothbrushCount >= 5 && GetAchievementsData.toothbrushCo
575     {
576         a35.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushBro
577         a35Txt.Text = "Eco Brush Bronze";
```

```
578     }
579     else if (GetAchievementsData.toothbrushCount >= 15 && GetAchievementsData.toothb
580     {
581         a35.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushSil
582         a35Txt.Text = "Eco Brush Silver";
583     }
584     else if (GetAchievementsData.toothbrushCount >= 25)
585     {
586         a35.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushGol
587         a35Txt.Text = "Eco Brush Gold";
588     }
589
590     if (GetAchievementsData.clothTowelCount >= 5 && GetAchievementsData.clothTowelCo
591     {
592         a36.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelBronze.p
593         a36Txt.Text = "Towel Bronze";
594     }
595     else if (GetAchievementsData.clothTowelCount >= 15 && GetAchievementsData.clothT
596     {
597         a36.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelSilver.p
598         a36Txt.Text = "Towel Silver";
599     }
600     else if (GetAchievementsData.clothTowelCount >= 25)
601     {
602         a36.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelGold.png
603         a36Txt.Text = "Towel Gold";
604     }
605
606     if (GetAchievementsData.reWaterCount >= 5 && GetAchievementsData.reWaterCount <
607     {
608         a37.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleBronze.png
609         a37Txt.Text = "Reusable Bottle Bronze";
610     }
611     else if (GetAchievementsData.reWaterCount >= 15 && GetAchievementsData.reWaterCo
612     {
613         a37.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleSilver.png
614         a37Txt.Text = "Reusable Bottle Silver";
615     }
616     else if (GetAchievementsData.reWaterCount >= 25)
617     {
618         a37.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleGold.png")
619         a37Txt.Text = "Reusable Bottle Gold";
620     }
621
622     if (GetAchievementsData.showerBucketCount >= 5 && GetAchievementsData.showerBuck
623     {
624         a38.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketBronze.png
625         a38Txt.Text = "Shower Bucket Bronze";
626     }
627     else if (GetAchievementsData.showerBucketCount >= 15 && GetAchievementsData.show
25)
628     {
629         a38.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketSilver.png
```



```
630         a38Txt.Text = "Shower Bucket Silver";
631     }
632     else if (GetAchievementsData.showerBucketCount >= 25)
633     {
634         a38.Source =
635         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketGold.png")
636         a38Txt.Text = "Shower Bucket Gold";
637     }
638     if (GetAchievementsData.airOutCount >= 5 && GetAchievementsData.airOutCount < 15
639     {
640         a39.Source =
641         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airBronze.png");
642         a39Txt.Text = "Ait Out Bronze";
643     }
644     else if (GetAchievementsData.airOutCount >= 15 && GetAchievementsData.airOutCoun
645     {
646         a39.Source =
647         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airSilver.png");
648         a39Txt.Text = "Ait Out Silver";
649     }
650     else if (GetAchievementsData.airOutCount >= 25)
651     {
652         a39.Source =
653         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airGold.png");
654         a39Txt.Text = "Ait Out Gold";
655     }
656     if (GetAchievementsData.toiletFlushCount >= 5 && GetAchievementsData.toiletFlush
657     {
658         a40.Source =
659         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushBronze.png")
660         a40Txt.Text = "Save Flush Bronze";
661     }
662     else if (GetAchievementsData.toiletFlushCount >= 15 && GetAchievementsData.toile
663     25)
664     {
665         a40.Source =
666         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushSilver.png")
667         a40Txt.Text = "Save Flush Silver";
668     }
669     else if (GetAchievementsData.toiletFlushCount >= 25)
670     {
671         a40.Source =
672         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushGold.png");
673         a40Txt.Text = "Save Flush Gold";
674     }
675     if (GetAchievementsData.nonHarmCount >= 5 && GetAchievementsData.nonHarmCount <
676     {
677         a41.Source =
678         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.nonHarmfulBronze.
679         a41Txt.Text = "Non Harmful Bronze";
680     }
681     else if (GetAchievementsData.nonHarmCount >= 15 && GetAchievementsData.nonHarmCo
682     {
683         a41.Source =
684         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.nonHarmfulSilver.
685         a41Txt.Text = "Non Harmful Silver";
686     }
687     else if (GetAchievementsData.nonHarmCount >= 25)
688     {
689         a41.Source =
```

```
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.nonHarmfulGold.png");
683     a41Txt.Text = "Non Harmful Gold";
684     }
685
686     if (GetAchievementsData.outsideCount >= 5 && GetAchievementsData.outsideCount <
687     {
688         a42.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.outsideBronze.png");
689         a42Txt.Text = "Go Outside Bronze";
690     }
691     else if (GetAchievementsData.outsideCount >= 15 && GetAchievementsData.outsideCo
692     {
693         a42.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.outsideSilver.png");
694         a42Txt.Text = "Go Outside Silver";
695     }
696     else if (GetAchievementsData.outsideCount >= 25)
697     {
698         a42.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.outsideGold.png");
699         a42Txt.Text = "Go Outside Gold";
700     }
701
702     if (GetAchievementsData.plantIntoHomeCount >= 5 && GetAchievementsData.plantInto
703     {
704         a43.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.plantBronze.png");
705         a43Txt.Text = "Home Plant Bronze";
706     }
707     else if (GetAchievementsData.plantIntoHomeCount >= 15 && GetAchievementsData.pla
< 25)
708     {
709         a43.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.plantSilver.png");
710         a43Txt.Text = "Home Plant Silver";
711     }
712     else if (GetAchievementsData.plantIntoHomeCount >= 25)
713     {
714         a43.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.plantGold.png");
715         a43Txt.Text = "Home Plant Gold";
716     }
717
718     if (GetAchievementsData.plantBushCount >= 5 && GetAchievementsData.plantBushCoun
719     {
720         a44.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.bushBronze.png");
721         a41Txt.Text = "Bush Planting Bronze";
722     }
723     else if (GetAchievementsData.plantBushCount >= 15 && GetAchievementsData.plantBu
724     {
725         a44.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.bushSilver.png");
726         a41Txt.Text = "Bush Planting Silver";
727     }
728     else if (GetAchievementsData.plantBushCount >= 25)
729     {
730         a44.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.bushGold.png");
731         a41Txt.Text = "Bush Planting Gold";
732     }
733
734     if (GetAchievementsData.campingCount >= 5 && GetAchievementsData.campingCount <
```

```
735     {
736         a45.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingBronze
737         a45Txt.Text = "Camping Bronze";
738     }
739     else if (GetAchievementsData.campingCount >= 15 && GetAchievementsData.campingCo
740     {
741         a45.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingSilver
742         a45Txt.Text = "Camping Silver";
743     }
744     else if (GetAchievementsData.campingCount >= 25)
745     {
746         a45.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingGold.p
747         a45Txt.Text = "Camping Gold";
748     }
749
750     if (GetAchievementsData.plantFlowerCount >= 5 && GetAchievementsData.plantFlower
751     {
752         a46.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerBronze.
753         a46Txt.Text = "Flower Planting Bronze";
754     }
755     else if (GetAchievementsData.plantFlowerCount >= 15 && GetAchievementsData.plant
25)
756     {
757         a46.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerSilver.
758         a46Txt.Text = "Flower Planting Silver";
759     }
760     else if (GetAchievementsData.plantFlowerCount >= 25)
761     {
762         a46.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerGold.pn
763         a46Txt.Text = "Flower Planting Gold";
764     }
765
766     if (GetAchievementsData.picnicCount >= 5 && GetAchievementsData.picnicCount < 15
767     {
768         a47.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicBronze.
769         a47Txt.Text = "Picnic Bronze";
770     }
771     else if (GetAchievementsData.picnicCount >= 15 && GetAchievementsData.picnicCoun
772     {
773         a47.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicSilver.
774         a47Txt.Text = "Picnic Silver";
775     }
776     else if (GetAchievementsData.picnicCount >= 25)
777     {
778         a47.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicGold.pn
779         a47Txt.Text = "Picnic Gold";
780     }
781
782     if (GetAchievementsData.scoopCount >= 5 && GetAchievementsData.scoopCount < 15)
783     {
784         a48.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.scoopBronze.p
785         a48Txt.Text = "Scoop Poop Bronze";
786     }
```

```
787     else if (GetAchievementsData.scoopCount >= 15 && GetAchievementsData.scoopCount
788     {
789         a48.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.scoopSilver.p
790         a48Txt.Text = "Scoop Poop Silver";
791     }
792     else if (GetAchievementsData.scoopCount >= 25)
793     {
794         a48.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.scoopGold.png
795         a48Txt.Text = "Scoop Poop Gold";
796     }
797
798     if (GetAchievementsData.plantTreeCount >= 5 && GetAchievementsData.plantTreeCoun
799     {
800         a49.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.treeBronze.pn
801         a49Txt.Text = "Tree Planting Bronze";
802     }
803     else if (GetAchievementsData.plantTreeCount >= 15 && GetAchievementsData.plantTr
804     {
805         a49.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.treeSilver.pn
806         a49Txt.Text = "Tree Planting Silver";
807     }
808     else if (GetAchievementsData.plantTreeCount >= 25)
809     {
810         a49.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.treeGold.png"
811         a49Txt.Text = "Tree Planting Gold";
812     }
813
814     if (GetAchievementsData.communityCount >= 5 && GetAchievementsData.communityCoun
815     {
816         a50.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.communityBro
817         a50Txt.Text = "Community Bronze";
818     }
819     else if (GetAchievementsData.communityCount >= 15 && GetAchievementsData.communi
820     {
821         a50.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.communitySil
822         a50Txt.Text = "Community Silver";
823     }
824     else if (GetAchievementsData.communityCount >= 25)
825     {
826         a50.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.communityGol
827         a50Txt.Text = "Community Gold";
828     }
829
830     if (GetAchievementsData.donateCount >= 5 && GetAchievementsData.donateCount < 15
831     {
832         a51.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.donateBronze
833         a51Txt.Text = "Donations Bronze";
834     }
835     else if (GetAchievementsData.donateCount >= 15 && GetAchievementsData.donateCoun
836     {
837         a51.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.donateSilver
838         a51Txt.Text = "Donations Silver";
839     }
```

```
840         else if (GetAchievementsData.donateCount >= 25)
841         {
842             a51.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.donateGold.p
843             a51Txt.Text = "Donations Gold";
844         }
845
846         if (GetAchievementsData.bioBinBagsCount >= 5 && GetAchievementsData.bioBinBagsCo
847         {
848             a52.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsBronze.png")
849             a52Txt.Text = "Bio Bin Bag Bronze";
850         }
851         else if (GetAchievementsData.bioBinBagsCount >= 15 && GetAchievementsData.bioBin
852         {
853             a52.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsSilver.png")
854             a52Txt.Text = "Bio Bin Bag Silver";
855         }
856         else if (GetAchievementsData.bioBinBagsCount >= 25)
857         {
858             a52.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsGold.png");
859             a52Txt.Text = "Bio Bin Bag Gold";
860         }
861
862         if (GetAchievementsData.billsCount >= 5 && GetAchievementsData.billsCount < 15)
863         {
864             a53.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsBronze.png"
865             a53Txt.Text = "Online Bills Bronze";
866         }
867         else if (GetAchievementsData.billsCount >= 15 && GetAchievementsData.billsCount
868         {
869             a53.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsSilver.png"
870             a53Txt.Text = "Online Bills Silver";
871         }
872         else if (GetAchievementsData.billsCount >= 25)
873         {
874             a53.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsGold.png");
875             a53Txt.Text = "Online Bills Gold";
876         }
877
878         if (GetAchievementsData.recyclingBinCount >= 5 && GetAchievementsData.recyclingB
879         {
880             a54.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binBronze.png");
881             a54Txt.Text = "Recycling Bronze";
882         }
883         else if (GetAchievementsData.recyclingBinCount >= 15 && GetAchievementsData.recy
25)
884         {
885             a54.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binSilver.png");
886             a54Txt.Text = "Recycling Silver";
887         }
888         else if (GetAchievementsData.recyclingBinCount >= 25)
889         {
890             a54.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binGold.png");
891             a54Txt.Text = "Recycling Gold";
```

```
892     }
893
894     if (GetAchievementsData.compostCount >= 5 && GetAchievementsData.compostCount <
895     {
896         a55.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.compostBronze.png");
897         a55Txt.Text = "Composting Bronze";
898     }
899     else if (GetAchievementsData.compostCount >= 15 && GetAchievementsData.compostCo
900     {
901         a55.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.compostSilver.png");
902         a55Txt.Text = "Composting Silver";
903     }
904     else if (GetAchievementsData.compostCount >= 25)
905     {
906         a55.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.compostGold.png");
907         a55Txt.Text = "Composting Gold";
908     }
909
910     if (GetAchievementsData.offElectronicsCount >= 5 && GetAchievementsData.offElect
15)
911     {
912         a56.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.offBronze.png");
913         a56Txt.Text = "Electronics Off Bronze";
914     }
915     else if (GetAchievementsData.offElectronicsCount >= 15 &&
GetAchievementsData.offElectronicsCount < 25)
916     {
917         a56.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.offSilver.png");
918         a56Txt.Text = "Electronics Off Silver";
919     }
920     else if (GetAchievementsData.offElectronicsCount >= 25)
921     {
922         a56.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.offGold.png");
923         a56Txt.Text = "Electronics Off Gold";
924     }
925
926     if (GetAchievementsData.paperCount >= 5 && GetAchievementsData.paperCount < 15)
927     {
928         a57.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.paperBronze.png");
929         a57Txt.Text = " Paper Bronze";
930     }
931     else if (GetAchievementsData.paperCount >= 15 && GetAchievementsData.paperCount
932     {
933         a57.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.paperSilver.png");
934         a57Txt.Text = " Paper Silver";
935     }
936     else if (GetAchievementsData.paperCount >= 25)
937     {
938         a57.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Work.paperGold.png");
939         a57Txt.Text = " Paper Gold";
940     }
941
942     if (GetAchievementsData.fixCount >= 5 && GetAchievementsData.fixCount < 15)
943     {
```



```
944         a58.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixBronze.png
945         a58Txt.Text = "Fixed Bronze";
946     }
947     else if (GetAchievementsData.fixCount >= 15 && GetAchievementsData.fixCount < 25
948     {
949         a58.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixSilver.png
950         a58Txt.Text = "Fixed Silver";
951     }
952     else if (GetAchievementsData.fixCount >= 25)
953     {
954         a58.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixGold.png")
955         a58Txt.Text = "Fixed Gold";
956     }
957
958     if (GetAchievementsData.efficientThermostatCount >= 1)
959     {
960         a59.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.thermos
961         a59Txt.Text = "Efficient Thermostat Set";
962     }
963
964     if (GetAchievementsData.insulateWaterCount >= 1)
965     {
966         a60.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.waterTa
967         a60Txt.Text = "Insulated Water Tank";
968     }
969
970     if (GetAchievementsData.isolateHomeCount >= 1)
971     {
972         a61.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.isolate
973         a61Txt.Text = "Insulated Home";
974     }
975
976     if (GetAchievementsData.ledLightBulbCount >= 1)
977     {
978         a62.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.ledBulb
979         a62Txt.Text = "Led Lights Installed";
980     }
981
982     if (GetAchievementsData.fridgeCount >= 1)
983     {
984         a63.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.turnDow
985         a63Txt.Text = "Turn Down The Fridge";
986     }
987
988     if (GetAchievementsData.draftSealCount >= 1)
989     {
990         a64.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.sealDra
991         a64Txt.Text = "Drafts Sealed";
992     }
993
994     if (GetAchievementsData.ductSealCount >= 1)
995     {
996         a65.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.sealDuc
```

```
997         a65Txt.Text = "Ducts Sealed";
998     }
999
1000     if (GetAchievementsData.solarPanelCount >= 1)
1001     {
1002         a66.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.solar.p
1003         a66Txt.Text = "Installed Solar Panel";
1004     }
1005
1006     if (GetAchievementsData.reusableCount >= 1)
1007     {
1008         a67.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Shopping.reWat
1009         a67Txt.Text = "Purchased Reusable Bottle";
1010     }
1011
1012     if (GetAchievementsData.reBatCount >= 1)
1013     {
1014         a68.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Shopping.reBat
1015         a68Txt.Text = "Purchased Reusable Batteries";
1016     }
1017
1018     if (GetAchievementsData.cisternCount >= 1)
1019     {
1020         a69.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.Cistern.
1021         a69Txt.Text = "Cistern Displacement System Installed";
1022     }
1023
1024     if (GetAchievementsData.rainBarrelCount >= 1)
1025     {
1026         a70.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.rainBarr
1027         a70Txt.Text = "Rain Barrel Set Up";
1028     }
1029
1030     if (GetAchievementsData.wSShowerHeadCount >= 1)
1031     {
1032         a71.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.showerHe
1033         a71Txt.Text = "Water Saving Shower Head Installed";
1034     }
1035
1036     if (GetAchievementsData.fruitGardenCount >= 1)
1037     {
1038         a72.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Outdoors.fruit
1039         a72Txt.Text = "Fruit Garden Set Up";
1040     }
1041
1042     if (GetAchievementsData.herbGardenCount >= 1)
1043     {
1044         a73.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Outdoors.herb.
1045         a73Txt.Text = "Herb Garden Set Up";
1046     }
1047
1048     if (GetAchievementsData.vegetableGardenCount >= 1)
1049     {
1050         a74.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Outdoors.veg.p
```



```
1051         a74Txt.Text = "Vegetable Garden Set Up";
1052     }
1053
1054     if (GetAchievementsData.birdFeederCount >= 1)
1055     {
1056         a75.Source =
1057         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Outdoors.birdF
1058         a75Txt.Text = "Bird Feeder Set Up";
1059     }
1060
1061     if (GetAchievementsData.createGroupCount >= 1)
1062     {
1063         a76.Source =
1064         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Community.crea
1065         a76Txt.Text = "Community Set Up";
1066     }
1067
1068     if (GetAchievementsData.groupCount >= 1)
1069     {
1070         a77.Source =
1071         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Community.join
1072         a77Txt.Text = "Joined A Community";
1073     }
1074
1075     if (GetAchievementsData.shareCount >= 1)
1076     {
1077         a78.Source =
1078         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Community.shar
1079         a78Txt.Text = "App Shared";
1080     }
1081
1082     if (GetAchievementsData.awarenessCount >= 1)
1083     {
1084         a79.Source =
1085         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Community.spre
1086         a79Txt.Text = "Awareness Spread";
1087     }
1088
1089     if (GetAchievementsData.setUpRecyclingBinCount >= 1)
1090     {
1091         a80.Source =
1092         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Waste.bins.png
1093         a80Txt.Text = "Recycling Bins Set Up";
1094     }
1095
1096     if (GetAchievementsData.remoteWorkCount >= 1)
1097     {
1098         a81.Source =
1099         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Work.remote.pn
1100         a81Txt.Text = "Working Remotely";
1101     }
1102 }
1103 }
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <pages:PopupPage x:Class="Application_Green_Quake.Views.ProfilePage.BadgePopUp"
3     xmlns="http://xamarin.com/schemas/2014/forms"
4     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5     xmlns:pages="clr-
namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
6     xmlns:local="clr-namespace:Application_Green_Quake.Models"
7     BackgroundColor="#002a1e">
8
9     <Grid VerticalOptions="FillAndExpand" RowSpacing="0" BackgroundColor="#002a1e">
10         <Grid.RowDefinitions>
11             <RowDefinition Height="*" />
12             <RowDefinition Height="2*" />
13             <RowDefinition Height="*" />
14         </Grid.RowDefinitions>
15
16         <StackLayout BackgroundColor="#002a1e"
17             VerticalOptions="End">
18             <Label x:Name="badgeHeading" Text="Unranked" FontSize="25" TextColor="White"
HorizontalTextAlignment="Center" />
19             <Label x:Name="badgeSubHeading" Text="" FontSize="15" TextColor="White"
HorizontalTextAlignment="Center" />
20         </StackLayout>
21
22         <Image Grid.Row="1"
23             Aspect="AspectFill"
24             BackgroundColor="#002a1e"
25             Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}"
26             x:Name="badge" />
27
28         <StackLayout Grid.Row="2"
29             BackgroundColor="#002a1e">
30             <Label x:Name="badgeTxt" Text="Next Rank: After 1 Action Log" FontSize="25"
TextColor="White" HorizontalTextAlignment="Center" />
31         </StackLayout>
32
33     </Grid>
34 </pages:PopupPage>
```

```
1  /*! \class The BadgePopUp View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the BadgePopUp View Class. This class is the popup that appears when
   badge is tapped.
7  *
8  */
9  using Xamarin.Forms;
10 using Xamarin.Forms.Xaml;
11
12 namespace Application_Green_Quake.Views.ProfilePage
13 {
14     [XamlCompilation(XamlCompilationOptions.Compile)]
15     public partial class BadgePopUp
16     {
17         int num = 0;
18         int badgeState= 0;
19         /** The BadgePopUp Constructor
20         @param number is used to specify which badge it is
21         @param stage is used to specify what stage the badge is in
22         */
23         public BadgePopUp(int number, int stage)
24         {
25             num = number;
26             badgeState = stage;
27             InitializeComponent();
28             OnAppearing();
29         }
30         /** This function is called before the page is displayed. It displays the correct
   information and badge based on what was passed into the
31         * constructor
32         */
33         protected override void OnAppearing()
34         {
35
36             if (num == 1 && badgeState == 1)
37             {
38                 badgeHeading.Text = "Habit Novice";
39                 badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
40                 badge.Source =
   ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsOne.png");
41                 badgeTxt.Text = "Next Rank: After 5 Action Logs";
42             }
43             else if (num == 1 && badgeState == 2)
44             {
45                 badgeHeading.Text = "Habit Apprentice";
46                 badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
47                 badge.Source =
   ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsTwo.png");
48                 badgeTxt.Text = "Next Rank: After 10 Action Logs";
49             }
50             else if (num == 1 && badgeState == 3)
51             {
52                 badgeHeading.Text = "Habit Adept";
53                 badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
54                 badge.Source =
   ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsThree.png");
55                 badgeTxt.Text = "Next Rank: After 25 Action Logs";
56             }
57         }
58     }
59 }
```

```
57     else if (num == 1 && badgeState == 4)
58     {
59         badgeHeading.Text = "Habit Expert";
60         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
61         badge.Source =
62 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFour.png");
63         badgeTxt.Text = "Next Rank: After 50 Action Logs";
64     }
65     else if (num == 1 && badgeState == 5)
66     {
67         badgeHeading.Text = "Habit Master";
68         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
69         badge.Source =
70 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFive.png");
71         badgeTxt.Text = "Next Rank: After 100 Action Logs";
72     }
73     else if (num == 1 && badgeState == 6)
74     {
75         badgeHeading.Text = "Habit Legend";
76         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
77         badge.Source =
78 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsSix.png");
79         badgeTxt.Text = "Max Rank Reached!";
80     }
81     if (num == 2 && badgeState == 1)
82     {
83         badgeHeading.Text = "Advanced Novice";
84         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
85         badge.Source =
86 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedOne.png");
87         badgeTxt.Text = "Next Rank: After 5 Action Logs";
88     }
89     else if (num == 2 && badgeState == 2)
90     {
91         badgeHeading.Text = "Advanced Apprentice";
92         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
93         badge.Source =
94 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedTwo.png");
95         badgeTxt.Text = "Next Rank: After 10 Action Logs";
96     }
97     else if (num == 2 && badgeState == 3)
98     {
99         badgeHeading.Text = "Advanced Habit Adept";
100        badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
101        badge.Source =
102 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedThree.png");
103        badgeTxt.Text = "Next Rank: After 25 Action Logs";
104     }
105     else if (num == 2 && badgeState == 4)
106     {
107        badgeHeading.Text = "Advanced Expert";
108        badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
109        badge.Source =
110 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFour.png");
111        badgeTxt.Text = "Next Rank: After 50 Action Logs";
112     }
113     else if (num == 2 && badgeState == 5)
114     {
115        badgeHeading.Text = "Advanced Master";
116        badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
117        badge.Source =
118 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFive.png");
```

```
112         badgeTxt.Text = "Next Rank: After 100 Action Logs";
113     }
114     else if (num == 2 && badgeState == 6)
115     {
116         badgeHeading.Text = "Advanced Legend";
117         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
118         badge.Source =
119         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedSix.png");
120         badgeTxt.Text = "Max Rank Reached!";
121     }
122     if (num == 3 && badgeState == 1)
123     {
124         badgeHeading.Text = "Community Novice";
125         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
126         badge.Source =
127         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityOne.png");
128         badgeTxt.Text = "Next Rank: After 5 Action Logs";
129     }
130     else if (num == 3 && badgeState == 2)
131     {
132         badgeHeading.Text = "Community Apprentice";
133         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
134         badge.Source =
135         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityTwo.png");
136         badgeTxt.Text = "Next Rank: After 10 Action Logs";
137     }
138     else if (num == 3 && badgeState == 3)
139     {
140         badgeHeading.Text = "Community Adept";
141         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
142         badge.Source =
143         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityThree.png");
144         badgeTxt.Text = "Next Rank: After 25 Action Logs";
145     }
146     else if (num == 3 && badgeState == 4)
147     {
148         badgeHeading.Text = "Community Expert";
149         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
150         badge.Source =
151         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFour.png");
152         badgeTxt.Text = "Next Rank: After 50 Action Logs";
153     }
154     else if (num == 3 && badgeState == 5)
155     {
156         badgeHeading.Text = "Community Master";
157         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
158         badge.Source =
159         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFive.png");
160         badgeTxt.Text = "Next Rank: After 100 Action Logs";
161     }
162     else if (num == 3 && badgeState == 6)
163     {
164         badgeHeading.Text = "Community Legend";
165         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
166         badge.Source =
167         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communitySix.png");
168         badgeTxt.Text = "Max Rank Reached!";
169     }
170     if (num == 4 && badgeState == 1)
171     {
```

```
167         badgeHeading.Text = "Energy Novice";
168         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
169         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyOne.png");
170         badgeTxt.Text = "Next Rank: After 5 Action Logs";
171     }
172     else if (num == 4 && badgeState == 2)
173     {
174         badgeHeading.Text = "Energy Apprentice";
175         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
176         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyTwo.png");
177         badgeTxt.Text = "Next Rank: After 10 Action Logs";
178     }
179     else if (num == 4 && badgeState == 3)
180     {
181         badgeHeading.Text = "Energy Adept";
182         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
183         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyThree.png");
184         badgeTxt.Text = "Next Rank: After 25 Action Logs";
185     }
186     else if (num == 4 && badgeState == 4)
187     {
188         badgeHeading.Text = "Energy Expert";
189         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
190         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFour.png");
191         badgeTxt.Text = "Next Rank: After 50 Action Logs";
192     }
193     else if (num == 4 && badgeState == 5)
194     {
195         badgeHeading.Text = "Energy Master";
196         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
197         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFive.png");
198         badgeTxt.Text = "Next Rank: After 100 Action Logs";
199     }
200     else if (num == 4 && badgeState == 6)
201     {
202         badgeHeading.Text = "Energy Legend";
203         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
204         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energySix.png");
205         badgeTxt.Text = "Max Rank Reached!";
206     }
207
208     if (num == 5 && badgeState == 1)
209     {
210         badgeHeading.Text = "Food And Drink Novice";
211         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
212         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDOne.png");
213         badgeTxt.Text = "Next Rank: After 5 Action Logs";
214     }
215     else if (num == 5 && badgeState == 2)
216     {
217         badgeHeading.Text = "Food And Drink Apprentice";
218         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
219         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDTwo.png");
220         badgeTxt.Text = "Next Rank: After 10 Action Logs";
221     }
```



```
222     else if (num == 5 && badgeState == 3)
223     {
224         badgeHeading.Text = "Food And Drink Adept";
225         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
226         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDThree.png");
227         badgeTxt.Text = "Next Rank: After 25 Action Logs";
228     }
229     else if (num == 5 && badgeState == 4)
230     {
231         badgeHeading.Text = "Food And Drink Expert";
232         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
233         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFour.png");
234         badgeTxt.Text = "Next Rank: After 50 Action Logs";
235     }
236     else if (num == 5 && badgeState == 5)
237     {
238         badgeHeading.Text = "Food And Drink Master";
239         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
240         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFive.png");
241         badgeTxt.Text = "Next Rank: After 100 Action Logs";
242     }
243     else if (num == 5 && badgeState == 6)
244     {
245         badgeHeading.Text = "Food And Drink Legend";
246         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
247         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDSix.png");
248         badgeTxt.Text = "Max Rank Reached!";
249     }
250
251     if (num == 6 && badgeState == 1)
252     {
253         badgeHeading.Text = "Home Novice";
254         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
255         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeOne.png");
256         badgeTxt.Text = "Next Rank: After 5 Action Logs";
257     }
258     else if (num == 6 && badgeState == 2)
259     {
260         badgeHeading.Text = "Home Apprentice";
261         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
262         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeTwo.png");
263         badgeTxt.Text = "Next Rank: After 10 Action Logs";
264     }
265     else if (num == 6 && badgeState == 3)
266     {
267         badgeHeading.Text = "Home Adept";
268         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
269         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeThree.png");
270         badgeTxt.Text = "Next Rank: After 25 Action Logs";
271     }
272     else if (num == 6 && badgeState == 4)
273     {
274         badgeHeading.Text = "Home Expert";
275         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
276         badge.Source =
```

```
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFour.png");
277     badgeTxt.Text = "Next Rank: After 50 Action Logs";
278 }
279     else if (num == 6 && badgeState == 5)
280     {
281         badgeHeading.Text = "Home Master";
282         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
283         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFive.png");
284         badgeTxt.Text = "Next Rank: After 100 Action Logs";
285     }
286     else if (num == 6 && badgeState == 6)
287     {
288         badgeHeading.Text = "Home Legend";
289         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
290         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeSix.png");
291         badgeTxt.Text = "Max Rank Reached!";
292     }
293
294     if (num == 7 && badgeState == 1)
295     {
296         badgeHeading.Text = "Outdoors Novice";
297         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
298         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsOne.png");
299         badgeTxt.Text = "Next Rank: After 5 Action Logs";
300     }
301     else if (num == 7 && badgeState == 2)
302     {
303         badgeHeading.Text = "Outdoors Apprentice";
304         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
305         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsTwo.png");
306         badgeTxt.Text = "Next Rank: After 10 Action Logs";
307     }
308     else if (num == 7 && badgeState == 3)
309     {
310         badgeHeading.Text = "Outdoors Adept";
311         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
312         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsThree.png");
313         badgeTxt.Text = "Next Rank: After 25 Action Logs";
314     }
315     else if (num == 7 && badgeState == 4)
316     {
317         badgeHeading.Text = "Outdoors Expert";
318         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
319         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFour.png");
320         badgeTxt.Text = "Next Rank: After 50 Action Logs";
321     }
322     else if (num == 7 && badgeState == 5)
323     {
324         badgeHeading.Text = "Outdoors Master";
325         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
326         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFive.png");
327         badgeTxt.Text = "Next Rank: After 100 Action Logs";
328     }
329     else if (num == 7 && badgeState == 6)
330     {
```



```
331         badgeHeading.Text = "Outdoors Legend";
332         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
333         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsSix.png");
334         badgeTxt.Text = "Max Rank Reached!";
335     }
336
337     if (num == 8 && badgeState == 1)
338     {
339         badgeHeading.Text = "Shopping Novice";
340         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
341         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingOne.png");
342         badgeTxt.Text = "Next Rank: After 5 Action Logs";
343     }
344     else if (num == 8 && badgeState == 2)
345     {
346         badgeHeading.Text = "Shopping Apprentice";
347         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
348         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingTwo.png");
349         badgeTxt.Text = "Next Rank: After 10 Action Logs";
350     }
351     else if (num == 8 && badgeState == 3)
352     {
353         badgeHeading.Text = "Shopping Adept";
354         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
355         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingThree.png");
356         badgeTxt.Text = "Next Rank: After 25 Action Logs";
357     }
358     else if (num == 8 && badgeState == 4)
359     {
360         badgeHeading.Text = "Shopping Expert";
361         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
362         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFour.png");
363         badgeTxt.Text = "Next Rank: After 50 Action Logs";
364     }
365     else if (num == 8 && badgeState == 5)
366     {
367         badgeHeading.Text = "Shopping Master";
368         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
369         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFive.png");
370         badgeTxt.Text = "Next Rank: After 100 Action Logs";
371     }
372     else if (num == 8 && badgeState == 6)
373     {
374         badgeHeading.Text = "Shopping Legend";
375         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
376         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingSix.png");
377         badgeTxt.Text = "Max Rank Reached!";
378     }
379
380     if (num == 9 && badgeState == 1)
381     {
382         badgeHeading.Text = "Travel Novice";
383         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
384         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelOne.png");
385         badgeTxt.Text = "Next Rank: After 5 Action Logs";
```

```
386     }
387     else if (num == 9 && badgeState == 2)
388     {
389         badgeHeading.Text = "Travel Apprentice";
390         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
391         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelTwo.png");
392         badgeTxt.Text = "Next Rank: After 10 Action Logs";
393     }
394     else if (num == 9 && badgeState == 3)
395     {
396         badgeHeading.Text = "Travel Adept";
397         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
398         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelThree.png");
399         badgeTxt.Text = "Next Rank: After 25 Action Logs";
400     }
401     else if (num == 9 && badgeState == 4)
402     {
403         badgeHeading.Text = "Travel Expert";
404         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
405         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFour.png");
406         badgeTxt.Text = "Next Rank: After 50 Action Logs";
407     }
408     else if (num == 9 && badgeState == 5)
409     {
410         badgeHeading.Text = "Travel Master";
411         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
412         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFive.png");
413         badgeTxt.Text = "Next Rank: After 100 Action Logs";
414     }
415     else if (num == 9 && badgeState == 6)
416     {
417         badgeHeading.Text = "Travel Legend";
418         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
419         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelSix.png");
420         badgeTxt.Text = "Max Rank Reached!";
421     }
422
423     if (num == 10 && badgeState == 1)
424     {
425         badgeHeading.Text = "Waste Novice";
426         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
427         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteOne.png");
428         badgeTxt.Text = "Next Rank: After 5 Action Logs";
429     }
430     else if (num == 10 && badgeState == 2)
431     {
432         badgeHeading.Text = "Waste Apprentice";
433         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
434         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteTwo.png");
435         badgeTxt.Text = "Next Rank: After 10 Action Logs";
436     }
437     else if (num == 10 && badgeState == 3)
438     {
439         badgeHeading.Text = "Waste Adept";
440         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
```

```
441         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteThree.png");
442         badgeTxt.Text = "Next Rank: After 25 Action Logs";
443     }
444     else if (num == 10 && badgeState == 4)
445     {
446         badgeHeading.Text = "Waste Expert";
447         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
448         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFour.png");
449         badgeTxt.Text = "Next Rank: After 50 Action Logs";
450     }
451     else if (num == 10 && badgeState == 5)
452     {
453         badgeHeading.Text = "Waste Master";
454         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
455         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFive.png");
456         badgeTxt.Text = "Next Rank: After 100 Action Logs";
457     }
458     else if (num == 10 && badgeState == 6)
459     {
460         badgeHeading.Text = "Waste Legend";
461         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
462         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteSix.png");
463         badgeTxt.Text = "Max Rank Reached!";
464     }
465
466     if (num == 11 && badgeState == 1)
467     {
468         badgeHeading.Text = "Water Novice";
469         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
470         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterOne.png");
471         badgeTxt.Text = "Next Rank: After 5 Action Logs";
472     }
473     else if (num == 11 && badgeState == 2)
474     {
475         badgeHeading.Text = "Water Apprentice";
476         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
477         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterTwo.png");
478         badgeTxt.Text = "Next Rank: After 10 Action Logs";
479     }
480     else if (num == 11 && badgeState == 3)
481     {
482         badgeHeading.Text = "Water Adept";
483         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
484         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterThree.png");
485         badgeTxt.Text = "Next Rank: After 25 Action Logs";
486     }
487     else if (num == 11 && badgeState == 4)
488     {
489         badgeHeading.Text = "Water Expert";
490         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
491         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFour.png");
492         badgeTxt.Text = "Next Rank: After 50 Action Logs";
493     }
494     else if (num == 11 && badgeState == 5)
495     {
```

```
496         badgeHeading.Text = "Water Master";
497         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
498         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFive.png");
499         badgeTxt.Text = "Next Rank: After 100 Action Logs";
500     }
501     else if (num == 11 && badgeState == 6)
502     {
503         badgeHeading.Text = "Water Legend";
504         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
505         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterSix.png");
506         badgeTxt.Text = "Max Rank Reached!";
507     }
508
509     if (num == 12 && badgeState == 1)
510     {
511         badgeHeading.Text = "Work Novice";
512         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
513         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workOne.png");
514         badgeTxt.Text = "Next Rank: After 5 Action Logs";
515     }
516     else if (num == 12 && badgeState == 2)
517     {
518         badgeHeading.Text = "Work Apprentice";
519         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
520         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workTwo.png");
521         badgeTxt.Text = "Next Rank: After 10 Action Logs";
522     }
523     else if (num == 12 && badgeState == 3)
524     {
525         badgeHeading.Text = "Work Adept";
526         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
527         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workThree.png");
528         badgeTxt.Text = "Next Rank: After 25 Action Logs";
529     }
530     else if (num == 12 && badgeState == 4)
531     {
532         badgeHeading.Text = "Work Expert";
533         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
534         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFour.png");
535         badgeTxt.Text = "Next Rank: After 50 Action Logs";
536     }
537     else if (num == 12 && badgeState == 5)
538     {
539         badgeHeading.Text = "Work Master";
540         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
541         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFive.png");
542         badgeTxt.Text = "Next Rank: After 100 Action Logs";
543     }
544     else if (num == 12 && badgeState == 6)
545     {
546         badgeHeading.Text = "Work Legend";
547         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
548         badge.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workSix.png");
549         badgeTxt.Text = "Max Rank Reached!";
550     }
```

```
551 |      }  
552 |    }  
553 | }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.ProfilePage.Badges"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6     <ContentPage.Content>
7         <ScrollView>
8             <Grid BackgroundColor="#002a1e" HorizontalOptions="Fill" VerticalOptions="Fill"
9             Padding="10,10,10,10">
10                <Grid.RowDefinitions>
11                    <RowDefinition Height="Auto"/>
12                    <RowDefinition Height="Auto"/>
13                    <RowDefinition Height="Auto"/>
14                </Grid.RowDefinitions>
15                <Grid.ColumnDefinitions>
16                    <ColumnDefinition Width="*"/>
17                    <ColumnDefinition Width="*"/>
18                    <ColumnDefinition Width="*"/>
19                    <ColumnDefinition Width="*"/>
20                </Grid.ColumnDefinitions>
21
22                <StackLayout Grid.Column="0" Grid.Row="0">
23                    <StackLayout.GestureRecognizers>
24                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpOne"/>
25                    </StackLayout.GestureRecognizers>
26                    <Frame CornerRadius="60" HorizontalOptions="Start"
27                    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
28                        <Image x:Name="a1" Source="{local:ImageResource
29                        Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
30                    </Frame>
31                    <Label x:Name="a1Txt" Text="Locked " TextColor="White"
32                    HorizontalTextAlignment="Center"/>
33                </StackLayout>
34
35                <StackLayout Grid.Column="1" Grid.Row="0">
36                    <StackLayout.GestureRecognizers>
37                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpTwo"/>
38                    </StackLayout.GestureRecognizers>
39                    <Frame CornerRadius="60" HorizontalOptions="Start"
40                    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
41                        <Image x:Name="a2" Source="{local:ImageResource
42                        Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
43                    </Frame>
44                    <Label x:Name="a2Txt" Text="Locked" TextColor="White"
45                    HorizontalTextAlignment="Center"/>
46                </StackLayout>
47
48                <StackLayout Grid.Column="2" Grid.Row="0">
49                    <StackLayout.GestureRecognizers>
50                        <TapGestureRecognizer Tapped="NavigateToBadgePopUpThree"/>
51                    </StackLayout.GestureRecognizers>
52                    <Frame CornerRadius="60" HorizontalOptions="Start"
53                    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
54                        <Image x:Name="a3" Source="{local:ImageResource
55                        Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
56                    </Frame>
57                    <Label x:Name="a3Txt" Text="Locked" TextColor="White"
58                    HorizontalTextAlignment="Center" />
59                </StackLayout>
60            </Grid>
61        </ScrollView>
62    </ContentPage.Content>
63</ContentPage>
```



```
53
54         <StackLayout Grid.Column="3" Grid.Row="0">
55             <StackLayout.GestureRecognizers>
56                 <TapGestureRecognizer Tapped="NavigateToBadgePopUpFour"/>
57             </StackLayout.GestureRecognizers>
58             <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
59                 <Image x:Name="a4" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
60             </Frame>
61
62             <Label x:Name="a4Txt" Text="Locked" TextColor="White"
HorizontalTextAlignment="Center" />
63         </StackLayout>
64
65         <StackLayout Grid.Column="0" Grid.Row="1">
66             <StackLayout.GestureRecognizers>
67                 <TapGestureRecognizer Tapped="NavigateToBadgePopUpFive"/>
68             </StackLayout.GestureRecognizers>
69             <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
70                 <Image x:Name="a5" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
71             </Frame>
72
73             <Label x:Name="a5Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center"/>
74         </StackLayout>
75
76         <StackLayout Grid.Column="1" Grid.Row="1">
77             <StackLayout.GestureRecognizers>
78                 <TapGestureRecognizer Tapped="NavigateToBadgePopUpSix"/>
79             </StackLayout.GestureRecognizers>
80             <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
81                 <Image x:Name="a6" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
82             </Frame>
83
84             <Label x:Name="a6Txt" Text="Locked" TextColor="White"
HorizontalTextAlignment="Center"/>
85         </StackLayout>
86
87         <StackLayout Grid.Column="2" Grid.Row="1">
88             <StackLayout.GestureRecognizers>
89                 <TapGestureRecognizer Tapped="NavigateToBadgePopUpSeven"/>
90             </StackLayout.GestureRecognizers>
91             <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
92                 <Image x:Name="a7" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
93             </Frame>
94
95             <Label x:Name="a7Txt" Text="Locked" TextColor="White"
HorizontalTextAlignment="Center" />
96         </StackLayout>
97
98         <StackLayout Grid.Column="3" Grid.Row="1">
99             <StackLayout.GestureRecognizers>
100                 <TapGestureRecognizer Tapped="NavigateToBadgePopUpEight"/>
101             </StackLayout.GestureRecognizers>
```

```
104         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
105             <Image x:Name="a8" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
106         </Frame>
107
108         <Label x:Name="a8Txt" Text="Locked" TextColor="White"
HorizontalTextAlignment="Center" />
109     </StackLayout>
110
111     <StackLayout Grid.Column="0" Grid.Row="2">
112         <StackLayout.GestureRecognizers>
113             <TapGestureRecognizer Tapped="NavigateToBadgePopUpNine"/>
114         </StackLayout.GestureRecognizers>
115         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
116             <Image x:Name="a9" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
117         </Frame>
118
119         <Label x:Name="a9Txt" Text="Locked " TextColor="White"
HorizontalTextAlignment="Center"/>
120     </StackLayout>
121
122     <StackLayout Grid.Column="1" Grid.Row="2">
123         <StackLayout.GestureRecognizers>
124             <TapGestureRecognizer Tapped="NavigateToBadgePopUpTen"/>
125         </StackLayout.GestureRecognizers>
126         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
127             <Image x:Name="a10" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
128         </Frame>
129
130         <Label x:Name="a10Txt" Text="Locked" TextColor="White"
HorizontalTextAlignment="Center"/>
131     </StackLayout>
132
133     <StackLayout Grid.Column="2" Grid.Row="2">
134         <StackLayout.GestureRecognizers>
135             <TapGestureRecognizer Tapped="NavigateToBadgePopUpEleven"/>
136         </StackLayout.GestureRecognizers>
137         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
138             <Image x:Name="a11" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
139         </Frame>
140
141         <Label x:Name="a11Txt" Text="Locked" TextColor="White"
HorizontalTextAlignment="Center" />
142     </StackLayout>
143
144     <StackLayout Grid.Column="3" Grid.Row="2">
145         <StackLayout.GestureRecognizers>
146             <TapGestureRecognizer Tapped="NavigateToBadgePopUpTwelve"/>
147         </StackLayout.GestureRecognizers>
148         <Frame CornerRadius="60" HorizontalOptions="Start"
VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
149             <Image x:Name="a12" Source="{local:ImageResource
Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand" />
150         </Frame>
151
152         <Label x:Name="a12Txt" Text="Locked" TextColor="White"
HorizontalTextAlignment="Center" />
```



```
153         </StackLayout>
154     </Grid>
155 </ScrollView>
156 </ContentPage.Content>
157 </ContentPage>
```

```

1  /!* \class The Badges View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the Badges View Class. This class is the class that displays all the
   Badges on the Badges page.
7  *
8  */
9  using Application_Green_Quake.ViewModels;
10 using Rg.Plugins.Popup.Services;
11 using System;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.ProfilePage
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class Badges : ContentPage
19     {
20         int habitsStage = 0;
21         int advancedStage = 0;
22         int communityStage = 0;
23         int energyStage = 0;
24         int foodDrinkStage = 0;
25         int homeStage = 0;
26         int outdoorsStage = 0;
27         int shoppingStage = 0;
28         int travelStage = 0;
29         int wasteStage = 0;
30         int waterStage = 0;
31         int workStage = 0;
32         public Badges()
33         {
34             InitializeComponent();
35             OnAppearing();
36         }
37         /** This function is called before the page is displayed. It displays the images as t
   criteria are met
38         */
39         protected override void OnAppearing()
40         {
41             if (GetBadgeData.habitsLog > 0 && GetBadgeData.habitsLog < 5)
42             {
43                 a1.Source =
44                 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsOne.png");
45                 a1Txt.Text = "Habits Novice";
46                 habitsStage = 1;
47             }
48             else if (GetBadgeData.habitsLog >= 5 && GetBadgeData.habitsLog < 10)
49             {
50                 a1.Source =
51                 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsTwo.png");
52                 a1Txt.Text = "Habits Apprentice";
53                 habitsStage = 2;
54             }
55             else if (GetBadgeData.habitsLog >= 10 && GetBadgeData.habitsLog < 25)
56             {
57                 a1.Source =
58                 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsThree.png");
59                 a1Txt.Text = "Habits Adept";

```

```
57         habitsStage = 3;
58     }
59     else if (GetBadgeData.habitsLog >= 25 && GetBadgeData.habitsLog < 50)
60     {
61         a1.Source =
62         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFour.png");
63         a1Txt.Text = "Habits Expert";
64         habitsStage = 4;
65     }
66     else if (GetBadgeData.habitsLog >= 50 && GetBadgeData.habitsLog < 100)
67     {
68         a1.Source =
69         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFive.png");
70         a1Txt.Text = "Habits Master";
71         habitsStage = 5;
72     }
73     else if (GetBadgeData.habitsLog >= 100)
74     {
75         a1.Source =
76         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsSix.png");
77         a1Txt.Text = "Habits Legend";
78         habitsStage = 6;
79     }
80
81     if (GetBadgeData.advancedLog > 0 && GetBadgeData.advancedLog < 5)
82     {
83         a2.Source =
84         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedOne.png");
85         a2Txt.Text = "Advanced Novice";
86         advancedStage = 1;
87     }
88     else if (GetBadgeData.advancedLog >= 5 && GetBadgeData.advancedLog < 10)
89     {
90         a2.Source =
91         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedTwo.png");
92         a2Txt.Text = "Advanced Apprentice";
93         advancedStage = 2;
94     }
95     else if (GetBadgeData.advancedLog >= 10 && GetBadgeData.advancedLog < 25)
96     {
97         a2.Source =
98         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedThree.png");
99         a2Txt.Text = "Advanced Adept";
100        advancedStage = 3;
101    }
102    else if (GetBadgeData.advancedLog >= 25 && GetBadgeData.advancedLog < 50)
103    {
104        a2.Source =
105        ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFour.png");
106        a2Txt.Text = "Advanced Expert";
107        advancedStage = 4;
108    }
109    else if (GetBadgeData.advancedLog >= 50 && GetBadgeData.advancedLog < 100)
110    {
111        a2.Source =
112        ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFive.png");
113        a2Txt.Text = "Advanced Master";
114        advancedStage = 5;
115    }
116    else if (GetBadgeData.advancedLog >= 100)
117    {
118        a2.Source =
119        ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedSix.png");
120        a2Txt.Text = "Advanced Legend";
121        advancedStage = 6;
122    }
```

```
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedSix.png");
112     a2Txt.Text = "Advanced Legend";
113     advancedStage = 6;
114 }
115
116     if (GetBadgeData.communityLog > 0 && GetBadgeData.communityLog < 5)
117     {
118         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityOne.png");
119         a3Txt.Text = "Community Novice";
120         communityStage = 1;
121     }
122     else if (GetBadgeData.communityLog >= 5 && GetBadgeData.communityLog < 10)
123     {
124         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityTwo.png");
125         a3Txt.Text = "Community Apprentice";
126         communityStage = 2;
127     }
128     else if (GetBadgeData.communityLog >= 10 && GetBadgeData.communityLog < 25)
129     {
130         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityThree.png");
131         a3Txt.Text = "Community Adept";
132         communityStage = 3;
133     }
134     else if (GetBadgeData.communityLog >= 25 && GetBadgeData.communityLog < 50)
135     {
136         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFour.png");
137         a3Txt.Text = "Community Expert";
138         communityStage = 4;
139     }
140     else if (GetBadgeData.communityLog >= 50 && GetBadgeData.communityLog < 100)
141     {
142         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFive.png");
143         a3Txt.Text = "Community Master";
144         communityStage = 5;
145     }
146     else if (GetBadgeData.communityLog >= 100)
147     {
148         a3.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communitySix.png");
149         a3Txt.Text = "Community Legend";
150         communityStage = 6;
151     }
152
153     if (GetBadgeData.energyLog > 0 && GetBadgeData.energyLog < 5)
154     {
155         a4.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyOne.png");
156         a4Txt.Text = "Energy Novice";
157         energyStage = 1;
158     }
159     else if (GetBadgeData.energyLog >= 5 && GetBadgeData.energyLog < 10)
160     {
161         a4.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyTwo.png");
162         a4Txt.Text = "Energy Apprentice";
163         energyStage = 2;
164     }
165     else if (GetBadgeData.energyLog >= 10 && GetBadgeData.energyLog < 25)
```

```
166     {
167         a4.Source =
168     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyThree.png");
169         a4Txt.Text = "Energy Adept";
170         energyStage = 3;
171     }
172     else if (GetBadgeData.energyLog >= 25 && GetBadgeData.energyLog < 50)
173     {
174         a4.Source =
175     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFour.png");
176         a4Txt.Text = "Energy Expert";
177         energyStage = 4;
178     }
179     else if (GetBadgeData.energyLog >= 50 && GetBadgeData.energyLog < 100)
180     {
181         a4.Source =
182     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFive.png");
183         a4Txt.Text = "Energy Master";
184         energyStage = 5;
185     }
186     else if (GetBadgeData.energyLog >= 100)
187     {
188         a4.Source =
189     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energySix.png");
190         a4Txt.Text = "Energy Legend";
191         energyStage = 6;
192     }
193     if (GetBadgeData.foodDrinkLog > 0 && GetBadgeData.foodDrinkLog < 5)
194     {
195         a5.Source =
196     ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDOne.png");
197         a5Txt.Text = "F & D Novice";
198         foodDrinkStage = 1;
199     }
200     else if (GetBadgeData.foodDrinkLog >= 5 && GetBadgeData.foodDrinkLog < 10)
201     {
202         a5.Source =
203     ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDTwo.png");
204         a5Txt.Text = "F & D Apprentice";
205         foodDrinkStage = 2;
206     }
207     else if (GetBadgeData.foodDrinkLog >= 10 && GetBadgeData.foodDrinkLog < 25)
208     {
209         a5.Source =
210     ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDThree.png");
211         a5Txt.Text = "F & D Adept";
212         foodDrinkStage = 3;
213     }
214     else if (GetBadgeData.foodDrinkLog >= 25 && GetBadgeData.foodDrinkLog < 50)
215     {
216         a5.Source =
217     ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFour.png");
218         a5Txt.Text = "F & D Expert";
219         foodDrinkStage = 4;
220     }
221     else if (GetBadgeData.foodDrinkLog >= 50 && GetBadgeData.foodDrinkLog < 100)
222     {
223         a5.Source =
224     ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFive.png");
225         a5Txt.Text = "F & D Master";
226         foodDrinkStage = 5;
227     }
228 }
```

```
220     else if (GetBadgeData.foodDrinkLog >= 100)
221     {
222         a5.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDSix.png");
223         a5Txt.Text = "F & D Legend";
224         foodDrinkStage = 6;
225     }
226
227     if (GetBadgeData.homeLog > 0 && GetBadgeData.homeLog < 5)
228     {
229         a6.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeOne.png");
230         a6Txt.Text = "Home Novice";
231         homeStage = 1;
232     }
233     else if (GetBadgeData.homeLog >= 5 && GetBadgeData.homeLog < 10)
234     {
235         a6.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeTwo.png");
236         a6Txt.Text = "Home Apprentice";
237         homeStage = 2;
238     }
239     else if (GetBadgeData.homeLog >= 10 && GetBadgeData.homeLog < 25)
240     {
241         a6.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeThree.png");
242         a6Txt.Text = "Home Adept";
243         homeStage = 3;
244     }
245     else if (GetBadgeData.homeLog >= 25 && GetBadgeData.homeLog < 50)
246     {
247         a6.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFour.png");
248         a6Txt.Text = "Home Expert";
249         homeStage = 4;
250     }
251     else if (GetBadgeData.homeLog >= 50 && GetBadgeData.homeLog < 100)
252     {
253         a6.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFive.png");
254         a6Txt.Text = "Home Master";
255         homeStage = 5;
256     }
257     else if (GetBadgeData.homeLog >= 100)
258     {
259         a6.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeSix.png");
260         a6Txt.Text = "Home Legend";
261         homeStage = 6;
262     }
263
264     if (GetBadgeData.outdoorsLog > 0 && GetBadgeData.outdoorsLog < 5)
265     {
266         a7.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsOne.png");
267         a7Txt.Text = "Outdoors Novice";
268         outdoorsStage = 1;
269     }
270     else if (GetBadgeData.outdoorsLog >= 5 && GetBadgeData.outdoorsLog < 10)
271     {
272         a7.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsTwo.png");
273         a7Txt.Text = "Outdoors Apprentice";
```

```
274         outdoorsStage = 2;
275     }
276     else if (GetBadgeData.outdoorsLog >= 10 && GetBadgeData.outdoorsLog < 25)
277     {
278         a7.Source = ImageSource.FromResource("Applica
tion_Green_Quake.Images.Badges.Outdoors.outdoorsThree.png");
279         a7Txt.Text = "Outdoors Adept";
280         outdoorsStage = 3;
281     }
282     else if (GetBadgeData.outdoorsLog >= 25 && GetBadgeData.outdoorsLog < 50)
283     {
284         a7.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFour.png");
285         a7Txt.Text = "Outdoors Expert";
286         outdoorsStage = 4;
287     }
288     else if (GetBadgeData.outdoorsLog >= 50 && GetBadgeData.outdoorsLog < 100)
289     {
290         a7.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFive.png");
291         a7Txt.Text = "Outdoors Master";
292         outdoorsStage = 5;
293     }
294     else if (GetBadgeData.outdoorsLog >= 100)
295     {
296         a7.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsSix.png");
297         a7Txt.Text = "Outdoors Legend";
298         outdoorsStage = 6;
299     }
300
301     if (GetBadgeData.shoppingLog > 0 && GetBadgeData.shoppingLog < 5)
302     {
303         a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingOne.png");
304         a8Txt.Text = "Shopping Novice";
305         shoppingStage = 1;
306     }
307     else if (GetBadgeData.shoppingLog >= 5 && GetBadgeData.shoppingLog < 10)
308     {
309         a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingTwo.png");
310         a8Txt.Text = "Shopping Apprentice";
311         shoppingStage = 2;
312     }
313     else if (GetBadgeData.shoppingLog >= 10 && GetBadgeData.shoppingLog < 25)
314     {
315         a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingThree.png");
316         a8Txt.Text = "Shopping Adept";
317         shoppingStage = 3;
318     }
319     else if (GetBadgeData.shoppingLog >= 25 && GetBadgeData.shoppingLog < 50)
320     {
321         a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFour.png");
322         a8Txt.Text = "Shopping Expert";
323         shoppingStage = 4;
324     }
325     else if (GetBadgeData.shoppingLog >= 50 && GetBadgeData.shoppingLog < 100)
326     {
327         a8.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFive.png");
```

```
328         a8Txt.Text = "Shopping Master";
329         shoppingStage = 5;
330     }
331     else if (GetBadgeData.shoppingLog >= 100)
332     {
333         a8.Source =
334         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingSix.png");
335         a8Txt.Text = "Shopping Legend";
336         shoppingStage = 6;
337     }
338     if (GetBadgeData.travelLog > 0 && GetBadgeData.travelLog < 5)
339     {
340         a9.Source =
341         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelOne.png");
342         a9Txt.Text = "Travel Novice";
343         travelStage = 1;
344     }
345     else if (GetBadgeData.travelLog >= 5 && GetBadgeData.travelLog < 10)
346     {
347         a9.Source =
348         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelTwo.png");
349         a9Txt.Text = "Travel Apprentice";
350         travelStage = 2;
351     }
352     else if (GetBadgeData.travelLog >= 10 && GetBadgeData.travelLog < 25)
353     {
354         a9.Source =
355         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelThree.png");
356         a9Txt.Text = "Travel Adept";
357         travelStage = 3;
358     }
359     else if (GetBadgeData.travelLog >= 25 && GetBadgeData.travelLog < 50)
360     {
361         a9.Source =
362         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFour.png");
363         a9Txt.Text = "Travel Expert";
364         travelStage = 4;
365     }
366     else if (GetBadgeData.travelLog >= 50 && GetBadgeData.travelLog < 100)
367     {
368         a9.Source =
369         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFive.png");
370         a9Txt.Text = "Travel Master";
371         travelStage = 5;
372     }
373     else if (GetBadgeData.travelLog >= 100)
374     {
375         a9.Source =
376         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelSix.png");
377         a9Txt.Text = "Travel Legend";
378         travelStage = 6;
379     }
380     if (GetBadgeData.wasteLog > 0 && GetBadgeData.wasteLog < 5)
381     {
382         a10.Source =
383         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteOne.png");
384         a10Txt.Text = "Waste Novice";
385         wasteStage = 1;
386     }
387     else if (GetBadgeData.wasteLog >= 5 && GetBadgeData.wasteLog < 10)
388     {
```



```
383         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteTwo.png");
384         a10Txt.Text = "Waste Apprentice";
385         wasteStage = 2;
386     }
387     else if (GetBadgeData.wasteLog >= 10 && GetBadgeData.wasteLog < 25)
388     {
389         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteThree.png");
390         a10Txt.Text = "Waste Adept";
391         wasteStage = 3;
392     }
393     else if (GetBadgeData.wasteLog >= 25 && GetBadgeData.wasteLog < 50)
394     {
395         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFour.png");
396         a10Txt.Text = "Waste Expert";
397         wasteStage = 4;
398     }
399     else if (GetBadgeData.wasteLog >= 50 && GetBadgeData.wasteLog < 100)
400     {
401         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFive.png");
402         a10Txt.Text = "Waste Master";
403         wasteStage = 5;
404     }
405     else if (GetBadgeData.wasteLog >= 100)
406     {
407         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteSix.png");
408         a10Txt.Text = "Waste Legend";
409         wasteStage = 6;
410     }
411
412     if (GetBadgeData.waterLog > 0 && GetBadgeData.waterLog < 5)
413     {
414         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterOne.png");
415         a11Txt.Text = "Water Novice";
416         waterStage = 1;
417     }
418     else if (GetBadgeData.waterLog >= 5 && GetBadgeData.waterLog < 10)
419     {
420         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterTwo.png");
421         a11Txt.Text = "Water Apprentice";
422         waterStage = 2;
423     }
424     else if (GetBadgeData.waterLog >= 10 && GetBadgeData.waterLog < 25)
425     {
426         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterThree.png");
427         a11Txt.Text = "Water Adept";
428         waterStage = 3;
429     }
430     else if (GetBadgeData.waterLog >= 25 && GetBadgeData.waterLog < 50)
431     {
432         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFour.png");
433         a11Txt.Text = "Water Expert";
434         waterStage = 4;
435     }
436     else if (GetBadgeData.waterLog >= 50 && GetBadgeData.waterLog < 100)
```

```
437     {
438         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFive.png");
439         a11Txt.Text = "Water Master";
440         waterStage = 5;
441     }
442     else if (GetBadgeData.waterLog >= 100)
443     {
444         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterSix.png");
445         a11Txt.Text = "Water Legend";
446         waterStage = 6;
447     }
448
449     if (GetBadgeData.workLog > 0 && GetBadgeData.workLog < 5)
450     {
451         a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workOne.png");
452         a12Txt.Text = "Work Novice";
453         workStage = 1;
454     }
455     else if (GetBadgeData.workLog >= 5 && GetBadgeData.workLog < 10)
456     {
457         a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workTwo.png");
458         a12Txt.Text = "Work Apprentice";
459         workStage = 2;
460     }
461     else if (GetBadgeData.workLog >= 10 && GetBadgeData.workLog < 25)
462     {
463         a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workThree.png");
464         a12Txt.Text = "Work Adept";
465         workStage = 3;
466     }
467     else if (GetBadgeData.workLog >= 25 && GetBadgeData.workLog < 50)
468     {
469         a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFour.png");
470         a12Txt.Text = "Work Expert";
471         workStage = 4;
472     }
473     else if (GetBadgeData.workLog >= 50 && GetBadgeData.workLog < 100)
474     {
475         a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFive.png");
476         a12Txt.Text = "Work Master";
477         workStage = 5;
478     }
479     else if (GetBadgeData.workLog >= 100)
480     {
481         a12.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workSix.png");
482         a12Txt.Text = "Work Legend";
483         workStage = 6;
484     }
485 }
486 /** This function displays a popup when the first badge is tapped
487 */
488 private async void NavigateToBadgePopUpOne(object sender, EventArgs e)
489 {
490     int number = 1;
491     await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, habitsStage));
```

```
492     }
493     /** This function displays a popup when the second badge is tapped
494     */
495     private async void NavigateToBadgePopUpTwo(object sender, EventArgs e)
496     {
497         int number = 2;
498         await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, advancedStage));
499     }
500     /** This function displays a popup when the third badge is tapped
501     */
502     private async void NavigateToBadgePopUpThree(object sender, EventArgs e)
503     {
504         int number = 3;
505         await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, communityStage));
506     }
507     /** This function displays a popup when the fourth badge is tapped
508     */
509     private async void NavigateToBadgePopUpFour(object sender, EventArgs e)
510     {
511         int number = 4;
512         await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, energyStage));
513     }
514     /** This function displays a popup when the fifth badge is tapped
515     */
516     private async void NavigateToBadgePopUpFive(object sender, EventArgs e)
517     {
518         int number = 5;
519         await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, foodDrinkStage));
520     }
521     /** This function displays a popup when the sixth badge is tapped
522     */
523     private async void NavigateToBadgePopUpSix(object sender, EventArgs e)
524     {
525         int number = 6;
526         await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, homeStage));
527     }
528     /** This function displays a popup when the seventh badge is tapped
529     */
530     private async void NavigateToBadgePopUpSeven(object sender, EventArgs e)
531     {
532         int number = 7;
533         await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, outdoorsStage));
534     }
535     /** This function displays a popup when the eight badge is tapped
536     */
537     private async void NavigateToBadgePopUpEight(object sender, EventArgs e)
538     {
539         int number = 8;
540         await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, shoppingStage));
541     }
542     /** This function displays a popup when the ninth badge is tapped
543     */
544     private async void NavigateToBadgePopUpNine(object sender, EventArgs e)
545     {
546         int number = 9;
547         await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, travelStage));
548     }
549     /** This function displays a popup when the tenth badge is tapped
550     */
551     private async void NavigateToBadgePopUpTen(object sender, EventArgs e)
552     {
```

```
553     int number = 10;
554     await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, wasteStage));
555 }
556 /** This function displays a popup when the eleventh badge is tapped
557 */
558 private async void NavigateToBadgePopUpEleven(object sender, EventArgs e)
559 {
560     int number = 11;
561     await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, waterStage));
562 }
563 /** This function displays a popup when the twelfth badge is tapped
564 */
565 private async void NavigateToBadgePopUpTwelve(object sender, EventArgs e)
566 {
567     int number = 12;
568     await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, workStage));
569 }
570 }
571 }
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.ProfilePage.TopTabProfile"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models">
6     <ContentPage.Content>
7         <ScrollView>
8             <StackLayout Spacing="0" Padding="0" BackgroundColor="#002a1e">
9                 <Frame Padding="0" CornerRadius="80" Margin="120,10,120,10"
10                    BackgroundColor="#33554b" HasShadow="False">
11                     <Frame Padding="0" CornerRadius="80" Margin="0,10,0,10"
12                        HeightRequest="90" WidthRequest="90" HorizontalOptions="CenterAndExpand"
13                        BackgroundColor="White">
14                         <Image x:Name="ProfilePic" Source="{local:ImageResource
15                            Application_Green_Quake.Images.user.png}" Aspect="AspectFill">
16                             <Image.GestureRecognizers>
17                                 <TapGestureRecognizer Tapped="ImageClicked"/>
18                             </Image.GestureRecognizers>
19                         </Image>
20                     </Frame>
21                 </Frame>
22             </StackLayout>
23
24             <StackLayout HeightRequest="100" VerticalOptions="Start"
25                HorizontalOptions="FillAndExpand" Spacing="0" BackgroundColor="#002a1e">
26                 <Label x:Name="Username" FontSize="28" HorizontalOptions="Center"
27                    VerticalOptions="Center" FontAttributes="Bold" TextColor="White"/>
28                 <Entry x:Name="Bio" Text="Bio" Completed="SaveText" TextColor="White"
29                    BackgroundColor="#002a1e" HorizontalOptions="FillAndExpand"
30                    VerticalOptions="CenterAndExpand" Margin="40,0,40,0" HorizontalTextAlignment="Center"
31                    PlaceholderColor="White"/>
32             </StackLayout>
33
34             <StackLayout Orientation="Horizontal" HeightRequest="50"
35                BackgroundColor="#002a1e" Padding="5">
36                 <StackLayout Spacing="0" BackgroundColor="#002a1e"
37                    Orientation="Horizontal" HorizontalOptions="Start">
38                     <Image Source="{local:ImageResource
39                        Application_Green_Quake.Images.TrophyCase.png}" WidthRequest="40" HeightRequest="40"
40                        HorizontalOptions="StartAndExpand" VerticalOptions="Center" Aspect="AspectFill">
41                         <Image.GestureRecognizers>
42                             <TapGestureRecognizer Tapped="NavigateToTrophyCase"/>
43                         </Image.GestureRecognizers>
44                     </Image>
45                     <Label FontSize="14" TextColor="White" Text="Trophy Case"
46                        HorizontalOptions="StartAndExpand" VerticalOptions="Center">
47                         <Label.GestureRecognizers>
48                             <TapGestureRecognizer Tapped="NavigateToTrophyCase"/>
49                         </Label.GestureRecognizers>
50                     </Label>
51                 </StackLayout>
52
53             <StackLayout Spacing="0" BackgroundColor="#002a1e"
54                Orientation="Horizontal" HorizontalOptions="EndAndExpand">
55                 <Image Source="{local:ImageResource
56                    Application_Green_Quake.Images.Achievements.png}" WidthRequest="40" HeightRequest="40"
57                    HorizontalOptions="StartAndExpand" VerticalOptions="Center" Aspect="AspectFill">
58                     <Image.GestureRecognizers>
59                         <TapGestureRecognizer Tapped="NavigateToAchievements"/>
60                     </Image.GestureRecognizers>
61                 </Image>
62                 <Label FontSize="14" TextColor="White" Text="Achievements"
63                    HorizontalOptions="StartAndExpand" VerticalOptions="Center">
64                     <Label.GestureRecognizers>
65                         <TapGestureRecognizer Tapped="NavigateToAchievements"/>
66                     </Label.GestureRecognizers>
67                 </Label>
68             </StackLayout>
69         </ScrollView>
70     </ContentPage.Content>
71 </ContentPage>

```

```
47         </Label.GestureRecognizers>
48     </Label>
49 </StackLayout>
50
51     <StackLayout Spacing="0" BackgroundColor="#002a1e"
Orientation="Horizontal" HorizontalOptions="EndAndExpand">
52         <Image Source="{local:ImageResource
Application_Green_Quake.Images.Badges.png}" WidthRequest="40" HeightRequest="40"
HorizontalOptions="StartAndExpand" VerticalOptions="Center" Aspect="AspectFill">
53             <Image.GestureRecognizers>
54                 <TapGestureRecognizer Tapped="NavigateToBadges"/>
55             </Image.GestureRecognizers>
56         </Image>
57         <Label FontSize="14" TextColor="White" Text="Badges"
HorizontalOptions="StartAndExpand" VerticalOptions="Center" >
58             <Label.GestureRecognizers>
59                 <TapGestureRecognizer Tapped="NavigateToBadges"/>
60             </Label.GestureRecognizers>
61         </Label>
62     </StackLayout>
63 </StackLayout>
64     <StackLayout VerticalOptions="FillAndExpand"
HorizontalOptions="FillAndExpand" Spacing="0" BackgroundColor="#002a1e">
65         <Label x:Name="theLevel" TextColor="White" Text="Level"
HorizontalOptions="Center" FontAttributes="Bold" FontSize="20"/>
66         <ProgressBar Margin="40,0,40,0" ProgressColor="Gold"
x:Name="progressbar"/>
67         <Image x:Name="mosiac" Source="{local:ImageResource
Application_Green_Quake.Images.Mosaics.lockedMosaics.jpg}" VerticalOptions="FillAndExpand"
HorizontalOptions="FillAndExpand" Aspect="Fill"/>
68     </StackLayout>
69 </StackLayout>
70 </ScrollView>
71 </ContentPage.Content>
72 </ContentPage>
```

```

1  /*! \class The TopTabProfile View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the TopTabProfile View Class. The progile page class.
7  *
8  */
9  using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using Firebase.Storage;
14 using Rg.Plugins.Popup.Services;
15 using System;
16 using Xamarin.Forms;
17 using Xamarin.Forms.Xaml;
18
19 namespace Application_Green_Quake.Views.ProfilePage
20 {
21     [XamlCompilation(XamlCompilationOptions.Compile)]
22     public partial class TopTabProfile : ContentPage
23     {
24         IAuth auth;
25         float progress = 0;
26         string bioInput = "";
27         float count = 0;
28
29         public TopTabProfile()
30         {
31             InitializeComponent();
32             auth = DependencyService.Get<IAuth>();
33             OnAppearing();
34         }
35         /** 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
36         * calulates the players level and progress and displays the progress to the next
   level on a progress bar.
37         */
38         protected override async void OnAppearing()
39         {
40             Username.Text = GetData.username;
41             try
42             {
43                 ProfilePic.Source = await new FirebaseStorage("application-green-
   quake.appspot.com")
44                     .Child(auth.GetUid())
45                     .Child("Profile.jpg")
46                     .GetDownloadUrlAsync();
47             }
48             catch (Exception e)
49             {
50                 Console.Write(e);
51             }
52
53             // Calculate the progress for the next level
54             progress = (float)GetData.points / 10;
55             progress = (int)((((decimal)progress % 1) * 10);
56
57             progress = progress / 10;

```



```
58
59     count = progress;
60     count = count * 10;
61
62     // Set the theLvl and animate the progress bar
63     theLevel.Text = "Lvl: " + GetData.lvl.ToString() + " Points: " +
count.ToString() + " /10";
64     await progressbar.ProgressTo(progress, 500, Easing.Linear);
65
66     if (GetData.lvl == 1)
67     {
68         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineOne.jpg");
69     }
70     else if (GetData.lvl == 2)
71     {
72         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineTwo.jpg");
73     }
74     else if (GetData.lvl == 3)
75     {
76         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineThree.jpg");
77     }
78     else if (GetData.lvl == 4)
79     {
80         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineFour.jpg");
81     }
82     else if (GetData.lvl == 5)
83     {
84         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1.jpg");
85     }
86     else if (GetData.lvl == 6)
87     {
88         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineOne.jpg");
89     }
90     else if (GetData.lvl == 7)
91     {
92         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineTwo.jpg");
93     }
94     else if (GetData.lvl == 8)
95     {
96         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineThree.jpg");
97     }
98     else if (GetData.lvl == 9)
99     {
100        mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineFour.jpg");
101    }
102    else if (GetData.lvl == 10)
103    {
104        mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2.jpg");
105    }
106    else if (GetData.lvl == 11)
107    {
108        mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineOne.jpg");
```



```
109     }
110     else if (GetData.lvl == 12)
111     {
112         mosaic.Source =
113         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineTwo.jpg");
114     }
115     else if (GetData.lvl == 13)
116     {
117         mosaic.Source =
118         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineThree.jpg");
119     }
120     else if (GetData.lvl == 14)
121     {
122         mosaic.Source =
123         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineFour.jpg");
124     }
125     else if (GetData.lvl == 15)
126     {
127         mosaic.Source =
128         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3.jpg");
129     }
130     else if (GetData.lvl == 16)
131     {
132         mosaic.Source =
133         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineOne.jpg");
134     }
135     else if (GetData.lvl == 17)
136     {
137         mosaic.Source =
138         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineTwo.jpg");
139     }
140     else if (GetData.lvl == 18)
141     {
142         mosaic.Source =
143         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineThree.jpg");
144     }
145     else if (GetData.lvl == 19)
146     {
147         mosaic.Source =
148         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineFour.jpg");
149     }
150     else if (GetData.lvl == 20)
151     {
152         mosaic.Source =
153         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4.jpg");
154     }
155     else if (GetData.lvl == 21)
156     {
157         mosaic.Source =
158         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineOne.jpg");
159     }
160     else if (GetData.lvl == 22)
161     {
162         mosaic.Source =
163         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineTwo.jpg");
164     }
165     else if (GetData.lvl == 23)
166     {
167         mosaic.Source =
168         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineThree.jpg");
169     }
170     else if (GetData.lvl == 24)
171     {
```

```
160         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineFour.jpg");
161     }
162     else if (GetData.lvl == 25)
163     {
164         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5.jpg");
165     }
166     else if (GetData.lvl == 26)
167     {
168         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineOne.jpg");
169     }
170     else if (GetData.lvl == 27)
171     {
172         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineTwo.jpg");
173     }
174     else if (GetData.lvl == 28)
175     {
176         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineThree.jpg");
177     }
178     else if (GetData.lvl == 29)
179     {
180         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineFour.jpg");
181     }
182     else if (GetData.lvl == 30)
183     {
184         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6.jpg");
185     }
186     else if (GetData.lvl == 31)
187     {
188         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineOne.jpg");
189     }
190     else if (GetData.lvl == 32)
191     {
192         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineTwo.jpg");
193     }
194     else if (GetData.lvl == 33)
195     {
196         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineThree.jpg");
197     }
198     else if (GetData.lvl == 34)
199     {
200         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineFour.jpg");
201     }
202     else if (GetData.lvl == 35)
203     {
204         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7.jpg");
205     }
206     else if (GetData.lvl == 36)
207     {
208         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineOne.jpg");
209     }
```

```
210         else if (GetData.lvl == 37)
211         {
212             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineTwo.jpg");
213         }
214         else if (GetData.lvl == 38)
215         {
216             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineThree.jpg");
217         }
218         else if (GetData.lvl == 39)
219         {
220             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineFour.jpg");
221         }
222         else if (GetData.lvl == 40)
223         {
224             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8.jpg");
225         }
226         else if (GetData.lvl == 41)
227         {
228             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineOne.jpg");
229         }
230         else if (GetData.lvl == 42)
231         {
232             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineTwo.jpg");
233         }
234         else if (GetData.lvl == 43)
235         {
236             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineThree.jpg");
237         }
238         else if (GetData.lvl == 44)
239         {
240             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineFour.jpg");
241         }
242         else if (GetData.lvl == 45)
243         {
244             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9.jpg");
245         }
246         else if (GetData.lvl == 46)
247         {
248             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineOne.jpg");
249         }
250         else if (GetData.lvl == 47)
251         {
252             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineTwo.jpg");
253         }
254         else if (GetData.lvl == 48)
255         {
256             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineThree.jpg");
257         }
258         else if (GetData.lvl == 49)
259         {
260             mosaic.Source =
```

```
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineFour.jpg");
261     }
262     else if (GetData.lvl == 50)
263     {
264         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10.jpg");
265     }
266     else if (GetData.lvl == 51)
267     {
268         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineOne.jpg");
269     }
270     else if (GetData.lvl == 52)
271     {
272         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineTwo.jpg");
273     }
274     else if (GetData.lvl == 53)
275     {
276         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineThree.jpg");
277     }
278     else if (GetData.lvl == 54)
279     {
280         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineFour.jpg");
281     }
282     else if (GetData.lvl == 55)
283     {
284         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11.jpg");
285     }
286     else if (GetData.lvl == 56)
287     {
288         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineOne.jpg");
289     }
290     else if (GetData.lvl == 57)
291     {
292         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineTwo.jpg");
293     }
294     else if (GetData.lvl == 58)
295     {
296         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineThree.jpg");
297     }
298     else if (GetData.lvl == 59)
299     {
300         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineFour.jpg");
301     }
302     else if (GetData.lvl == 60)
303     {
304         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12.jpg");
305     }
306     else if (GetData.lvl == 61)
307     {
308         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineOne.jpg");
309     }
310     else if (GetData.lvl == 62)
```

```
311     {
312         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineTwo.jpg");
313     }
314     else if (GetData.lvl == 63)
315     {
316         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineThree.jpg");
317     }
318     else if (GetData.lvl == 64)
319     {
320         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineFour.jpg");
321     }
322     else if (GetData.lvl == 65)
323     {
324         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13.jpg");
325     }
326     else if (GetData.lvl == 66)
327     {
328         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineOne.jpg");
329     }
330     else if (GetData.lvl == 67)
331     {
332         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineTwo.jpg");
333     }
334     else if (GetData.lvl == 68)
335     {
336         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineThree.jpg");
337     }
338     else if (GetData.lvl == 69)
339     {
340         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineFour.jpg");
341     }
342     else if (GetData.lvl == 70)
343     {
344         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14.jpg");
345     }
346     else if (GetData.lvl == 71)
347     {
348         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineOne.jpg");
349     }
350     else if (GetData.lvl == 72)
351     {
352         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineTwo.jpg");
353     }
354     else if (GetData.lvl == 73)
355     {
356         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineThree.jpg");
357     }
358     else if (GetData.lvl == 74)
359     {
360         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineFour.jpg");
```

```
361     }
362     else if (GetData.lvl == 75)
363     {
364         mosaic.Source =
365         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15.jpg");
366     }
367     else if (GetData.lvl == 76)
368     {
369         mosaic.Source =
370         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineOne.jpg");
371     }
372     else if (GetData.lvl == 77)
373     {
374         mosaic.Source =
375         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineTwo.jpg");
376     }
377     else if (GetData.lvl == 78)
378     {
379         mosaic.Source =
380         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineThree.jpg");
381     }
382     else if (GetData.lvl == 79)
383     {
384         mosaic.Source =
385         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineFour.jpg");
386     }
387     else if (GetData.lvl == 80)
388     {
389         mosaic.Source =
390         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16.jpg");
391     }
392     else if (GetData.lvl == 81)
393     {
394         mosaic.Source =
395         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineOne.jpg");
396     }
397     else if (GetData.lvl == 82)
398     {
399         mosaic.Source =
400         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineTwo.jpg");
401     }
402     else if (GetData.lvl == 83)
403     {
404         mosaic.Source =
405         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineThree.jpg");
406     }
407     else if (GetData.lvl == 84)
408     {
409         mosaic.Source =
410         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineFour.jpg");
411     }
412     else if (GetData.lvl == 85)
413     {
414         mosaic.Source =
415         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17.jpg");
416     }
417     else if (GetData.lvl == 86)
418     {
419         mosaic.Source =
420         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineOne.jpg");
421     }
422     else if (GetData.lvl == 87)
423     {
```

```
412         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineTwo.jpg");
413     }
414     else if (GetData.lvl == 88)
415     {
416         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineThree.jpg");
417     }
418     else if (GetData.lvl == 89)
419     {
420         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineFour.jpg");
421     }
422     else if (GetData.lvl == 90)
423     {
424         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18.jpg");
425     }
426     else if (GetData.lvl == 91)
427     {
428         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineOne.jpg");
429     }
430     else if (GetData.lvl == 92)
431     {
432         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineTwo.jpg");
433     }
434     else if (GetData.lvl == 93)
435     {
436         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineThree.jpg");
437     }
438     else if (GetData.lvl == 94)
439     {
440         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineFour.jpg");
441     }
442     else if (GetData.lvl == 95)
443     {
444         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19.jpg");
445     }
446     else if (GetData.lvl == 96)
447     {
448         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineOne.jpg");
449     }
450     else if (GetData.lvl == 97)
451     {
452         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineTwo.jpg");
453     }
454     else if (GetData.lvl == 98)
455     {
456         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineThree.jpg");
457     }
458     else if (GetData.lvl == 99)
459     {
460         mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineFour.jpg");
461     }
```



```
462         else if (GetData.lvl == 100)
463         {
464             mosaic.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20.jpg");
465         }
466
467         GetData data = new GetData();
468         data.SetLvl();
469     }
470     /** This function navigates to UploadImagePopUp
471     */
472     private async void ImageClicked(object sender, EventArgs e)
473     {
474         await PopupNavigation.Instance.PushAsync(new UploadImagePopUp());
475     }
476     /** This function navigates to TrophyCase at tab 0
477     */
478     private async void NavigateToTrophyCase(object sender, EventArgs e)
479     {
480         await Navigation.PushAsync(new TrophyCase(0));
481     }
482     /** This function navigates to TrophyCase at tab 1
483     */
484     private async void NavigateToAchievements(object sender, EventArgs e)
485     {
486         await Navigation.PushAsync(new TrophyCase(1));
487     }
488     /** This function navigates to TrophyCase at tab 2
489     */
490     private async void NavigateToBadges(object sender, EventArgs e)
491     {
492         await Navigation.PushAsync(new TrophyCase(2));
493     }
494     /** This function saves the users bio
495     */
496     private async void SaveText(object sender, EventArgs e)
497     {
498         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
499
500         bioInput = Bio.Text;
501
502         await firebaseClient
503             .Child("users")
504             .Child(auth.GetUid())
505             .PutAsync(new Users() {username = GetData.username ,bio = bioInput,
nation = GetData.nation});
506     }
507 }
508 }
```



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
3     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"
4     xmlns:models="clr-namespace:Application_Green_Quake.Models;assembly=Application
Green Quake"
5     x:Class="Application_Green_Quake.Views.ProfilePage.TrophyCase"
6     NavigationPage.HasBackButton="False"
7     NavigationPage.HasNavigationBar="False">
8     <ContentPage Title="Trophies">
9         <ContentPage.Content>
10             <ScrollView>
11                 <Grid HorizontalOptions="Fill" VerticalOptions="Fill" Padding="10,30,10,10"
Row="20">
12                     <Grid.RowDefinitions>
13                         <RowDefinition Height="*"/>
14                         <RowDefinition Height="*"/>
15                         <RowDefinition Height="*"/>
16                         <RowDefinition Height="*"/>
17                     </Grid.RowDefinitions>
18                     <Grid.ColumnDefinitions>
19                         <ColumnDefinition Width="*"/>
20                     </Grid.ColumnDefinitions>
21                     <StackLayout Grid.Row="0" >
22                         <Image x:Name="t1"
23                             Source="{models:ImageResource
Application_Green_Quake.Images.lockOne.png}"
24                             HorizontalOptions="CenterAndExpand">
25                         </Image>
26                         <Label x:Name="t1Txt" Text="1000 Points to unlock" TextColor="Black"
HorizontalTextAlignment="Center"/>
27                     </StackLayout>
28                     <StackLayout Grid.Row="1">
29                         <Image x:Name="t2"
30                             Source="{models:ImageResource
Application_Green_Quake.Images.lockOne.png}"
31                             HorizontalOptions="CenterAndExpand">
32                         </Image>
33                         <Label x:Name="t2Txt" Text="500 Points to unlock" TextColor="Black"
HorizontalTextAlignment="Center"/>
34                     </StackLayout>
35                     <StackLayout Grid.Row="2">
36                         <Image x:Name="t3"
37                             Source="{models:ImageResource
Application_Green_Quake.Images.lockOne.png}"
38                             HorizontalOptions="CenterAndExpand">
39                         </Image>
40                         <Label x:Name="t3Txt" Text="250 Points to unlock" TextColor="Black"
HorizontalTextAlignment="Center"/>
41                     </StackLayout>
42                     <StackLayout Grid.Row="3">
43                         <Image x:Name="t4"
44                             Source="{models:ImageResource
Application_Green_Quake.Images.lockOne.png}"
45                             HorizontalOptions="CenterAndExpand">
46                         </Image>
47                         <Label x:Name="t4Txt" Text="100 Points to unlock" TextColor="Black"
HorizontalTextAlignment="Center"/>
48                     </StackLayout>
49                 </Grid>
50             </ScrollView>
51         </ContentPage.Content>
```

```
52 | </ContentPage>
53 |
54 | <views1:Achievements Title="Achievements"></views1:Achievements>
55 | <views1:Badges Title="Badges"></views1:Badges>
56 | </TabbedPage>
```

```

1  /!* \class The TrophyCase View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the TrophyCase View Class. This class is the class that displays all
   the trophies on the trophy page.
7  *
8  */
9  using Application_Green_Quake.ViewModels;
10 using Xamarin.Forms;
11 using Xamarin.Forms.Xaml;
12
13 namespace Application_Green_Quake.Views.ProfilePage
14 {
15     [XamlCompilation(XamlCompilationOptions.Compile)]
16     public partial class TrophyCase : TabbedPage
17     {
18         /** The constructor for Main menu
19         @param tab supplied to tell the class which tabbed page to display.
20         */
21         public TrophyCase(int tab)
22         {
23             InitializeComponent();
24             CurrentPage = Children[tab];
25             OnAppearing();
26         }
27         /** This function is called before the page is displayed. It displays the image as
   the criteria is met
28         */
29         protected override void OnAppearing()
30         {
31             if (GetData.points >= 1000)
32             {
33                 t1.Source =
34                 ImageSource.FromResource("Application_Green_Quake.Images.Trophies.diamond.png");
35                 t1Txt.Text = "Diamond Trophy";
36             }
37             if (GetData.points >= 500)
38             {
39                 t2.Source =
40                 ImageSource.FromResource("Application_Green_Quake.Images.Trophies.gold.png");
41                 t2Txt.Text = "Gold Trophy";
42             }
43             if (GetData.points >= 250)
44             {
45                 t3.Source =
46                 ImageSource.FromResource("Application_Green_Quake.Images.Trophies.silver.png");
47                 t3Txt.Text = "Silver Trophy";
48             }
49             if (GetData.points >= 100)
50             {
51                 t4.Source =
52                 ImageSource.FromResource("Application_Green_Quake.Images.Trophies.bronze.png");
53                 t4Txt.Text = "Bronze Trophy";
54             }
55         }
56     }
57 }

```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <pages:PopupPage x:Class="Application_Green_Quake.Views.ProfilePage.UploadImagePopUp"
3     xmlns="http://xamarin.com/schemas/2014/forms"
4     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5     xmlns:pages="clr-
6     namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
7     BackgroundColor="#002a1e">
8     <StackLayout HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" >
9         <Image x:Name="ChosenImage" HeightRequest="200"></Image>
10        <Button x:Name="Take" TextColor="White" BackgroundColor="#50C878" Text="Take Photo"
11        Clicked="CapturePhotoClicked"></Button>
12        <Button x:Name="Pick" TextColor="White" BackgroundColor="#50C878" Text="Pick From
13        Storage" Clicked="UplaodPhotoClicked"></Button>
14        <Button x:Name="Save" TextColor="White" BackgroundColor="#50C878" Text="Save"
15        Clicked="storeImageClicked"></Button>
16    </StackLayout>
17</pages:PopupPage>
```

```
1 /!* \class The UploadImagePopUp View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the UploadImagePopUp View Class. This class is the popup that
   appears to upload a new profile picture.
7 *
8 */
9 using Plugin.Media;
10 using System;
11 using System.Diagnostics;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14 using System.IO;
15 using System.Threading.Tasks;
16 using Acr.UserDialogs;
17 using Firebase.Storage;
18 using Plugin.Media.Abstractions;
19 using Rg.Plugins.Popup.Services;
20
21 namespace Application_Green_Quake.Views.ProfilePage
22 {
23     [XamlCompilation(XamlCompilationOptions.Compile)]
24     public partial class UploadImagePopUp
25     {
26         IAuth auth;
27         public MediaFile File { get; set; }
28
29         public UploadImagePopUp()
30         {
31             InitializeComponent();
32             auth = DependencyService.Get<IAuth>();
33         }
34         /** This function enables the capturing of an image.
35         */
36         private async void CapturePhotoClicked(object sender, System.EventArgs e)
37         {
38             await CrossMedia.Current.Initialize();
39
40             if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakePhotoSupported)
41             {
42                 await DisplayAlert("No Camera", "No Camera Detected", "OK");
43                 return;
44             }
45
46             File = await CrossMedia.Current.TakePhotoAsync(
47                 new StoreCameraMediaOptions
48                 {
49                     SaveToAlbum = true
50                 }
51             );
52             if (File == null)
53                 return;
54
55             ChosenImage.Source = ImageSource.FromStream(() =>
56             {
57                 var stream = File.GetStream();
58                 return stream;
```

```
59     });
60 }
61
62 /** This function enables the uploading of an image.
63 */
64 private async void UplaodPhotoClicked(object sender, System.EventArgs e)
65 {
66     await CrossMedia.Current.Initialize();
67     try
68     {
69         File = await CrossMedia.Current.PickPhotoAsync(new PickMediaOptions
70         {
71             PhotoSize = PhotoSize.Medium
72         });
73         if (File == null)
74             return;
75         ChosenImage.Source = ImageSource.FromStream(() =>
76         {
77             var imageSteram = File.GetStream();
78             return imageSteram;
79         });
80     }
81     catch (Exception ex)
82     {
83         Debug.WriteLine(ex.Message);
84     }
85 }
86
87 /** This function enables the storing of an image.
88 */
89 private async void storeImageClicked(object sender, System.EventArgs e)
90 {
91     UserDialogs.Instance.ShowLoading();
92     try
93     {
94         {
95             await StoreImages(File.GetStream());
96         }
97
98         async Task<string> StoreImages(Stream imageStream)
99         {
100             var stroageImage = await new FirebaseStorage("application-green-
101 quake.appspot.com")
102                 .Child(auth.GetUid())
103                 .Child("Profile.jpg")
104                 .PutAsync(imageStream);
105             string imgurl = stroageImage;
106             return imgurl;
107         }
108     }
109     catch (Exception ex)
110     {
111         Debug.WriteLine(ex.Message);
112     }
113
114     await Navigation.PushAsync(new MainMenu(2));
115     await PopupNavigation.Instance.PopAsync(true);
116     // Hide the loading screen
117     UserDialogs.Instance.HideLoading();
118 }
```

```
119 | }  
120 | }
```

```
1 /*! \class The IAuth Interface
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the IAuth Interface. It is the interface between the cross platform
   code and the native code.
7 *
8 */
9 using System;
10 using System.Threading.Tasks;
11
12 namespace Application_Green_Quake
13 {
14     public interface IAuth
15     {
16         Task<String> LoginWithEmailAndPassword(string email, string password);
17
18         Task<String> SignUpWithEmailAndPassword(string email, string password);
19
20         Task ResetPassword(string email);
21
22         bool SignOut();
23
24         bool IsSignIn();
25
26         string GetUid();
27     }
28 }
```



```
1 /*! \class The IAuth Interface
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the IAuth Interface. It is the interface between the cross platform
   code and the native code.
7 *
8 */
9 using System;
10 using System.Threading.Tasks;
11
12 namespace Application_Green_Quake
13 {
14     public interface IAuth
15     {
16         Task<String> LoginWithEmailAndPassword(string email, string password);
17
18         Task<String> SignUpWithEmailAndPassword(string email, string password);
19
20         Task ResetPassword(string email);
21
22         bool SignOut();
23
24         bool IsSignIn();
25
26         string GetUid();
27     }
28 }
```

```
1 using Xamarin.Forms.Xaml;  
2 [assembly: XamlCompilation(XamlCompilationOptions.Compile)]  
3 namespace Application_Green_Quake  
4 {  
5  
6 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <Application xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.App">
5     <Application.Resources>
6         <!-- Colors -->
7         <Color x:Key="AppPrimaryColor">Red</Color>
8         <Color x:Key="AppBackgroundColor">White</Color>
9         <Color x:Key="PrimaryColor">Black</Color>
10        <Color x:Key="SecondaryColor">#50C878</Color>
11        <Color x:Key="TertiaryColor">Yellow</Color>
12
13        <!-- Implicit styles -->
14        <Style TargetType="ContentPage"
15            ApplyToDerivedTypes="True">
16            <Setter Property="BackgroundColor"
17                Value="{StaticResource AppBackgroundColor}" />
18
19        </Style>
20
21        <Style TargetType="TabbedPage"
22            ApplyToDerivedTypes="True">
23            <Setter Property="BarBackgroundColor"
24                Value="{StaticResource SecondaryColor}" />
25            <Setter Property="UnselectedTabColor"
26                Value="{StaticResource PrimaryColor}" />
27            <Setter Property="SelectedTabColor"
28                Value="{StaticResource AppBackgroundColor}" />
29        </Style>
30
31        <Style TargetType="NavigationPage"
32            ApplyToDerivedTypes="True">
33            <Setter Property="BarBackgroundColor"
34                Value="{StaticResource SecondaryColor}" />
35        </Style>
36    </Application.Resources>
37 </Application>
```

```
1 /!* \mainpage The App Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946, c002289
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the App Class. This is the class that gets loaded first on app launch.
7 *
8 */
9 using Application_Green_Quake.ViewModels;
10 using Microsoft.AppCenter;
11 using Microsoft.AppCenter.Analytics;
12 using Microsoft.AppCenter.Crashes;
13 using Application_Green_Quake.Views;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake
17 {
18     public partial class App : Application
19     {
20         IAuth auth;
21         public App()
22         {
23             //Register Syncfusion license
24             Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense("NDM10Tg4QDMxMzkyZTMxMmUzME5wen
25             GetData set = new GetData();
26             set.SetLvl();
27             InitializeComponent();
28             auth = DependencyService.Get<IAuth>();
29
30             //If the user is signed in navigate to the main menu
31             if (auth.IsSignIn())
32             {
33                 MainPage = new NavigationPage(new MainMenu());
34             }
35             //If the users is not signed in navigate to the login screen
36             else
37             {
38                 MainPage = new NavigationPage(new MainPage());
39             }
40         }
41
42         protected override void OnStart()
43         {
44             GetData data = new GetData();
45             data.SetLvl();
46             AppCenter.Start("android=87250b90-3ea3-429d-ac0b-7e47e6cd70ac;" +
47                 "uwp={Your UWP App secret here};" +
48                 "ios={Your iOS App secret here}",
49                 typeof(Analytics), typeof(Crashes));
50         }
51
52         protected override void OnSleep()
53         {
54         }
55
56         protected override void OnResume()
57         {
58             GetData data = new GetData();
59             data.SetLvl();
60         }
61     }
62 }
```

```
61 | }  
62 | }
```

```

1  /*! \class The AuthDroid Native Android Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * 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
7  * sign up to get their uid and to get a password rest email.
8  *
9  */
10 using System;
11 using System.Threading.Tasks;
12 using Xamarin.Forms;
13 using Application_Green_Quake.Droid;
14 using Firebase.Auth;
15
16 [assembly: Dependency(typeof(AuthDroid))]
17 namespace Application_Green_Quake.Droid
18 {
19     public class AuthDroid : IAuth
20     {
21         /** This function checks if a user is signed in.
22         */
23         public bool IsSignIn()
24         {
25             var user = FirebaseAuth.Instance.CurrentUser;
26             return user != null;
27         }
28         /** This function logs a user into the application
29         */
30         public async Task<string> LoginWithEmailAndPassword(string email, string password)
31         {
32             try
33             {
34                 var user = await
35 FirebaseAuth.Instance.SignInWithEmailAndPasswordAsync(email, password);
36                 var token = user.User.Uid;
37                 return token;
38             }
39             catch (FirebaseAuthInvalidUserException e)
40             {
41                 e.PrintStackTrace();
42                 return string.Empty;
43             }
44             catch (FirebaseAuthInvalidCredentialsException e)
45             {
46                 e.PrintStackTrace();
47                 return string.Empty;
48             }
49         }
50         /** This function signs a user out of the application.
51         */
52         public bool SignOut()
53         {
54             try
55             {
56                 FirebaseAuth.Instance.SignOut();
57                 return true;
58             }

```

```
59     catch (Exception e)
60     {
61         Console.WriteLine(e);
62         return false;
63     }
64 }
65 /** This function allows a user to sign up into the app.
66 */
67 public async Task<string> SignUpWithEmailAndPassword(string email, string password)
68 {
69     try
70     {
71         var newUser = await
FirebaseAuth.Instance.CreateUserWithEmailAndPasswordAsync(email, password);
72         var token = newUser.User.Uid;
73         return token;
74     }
75     catch (FirebaseAuthUserCollisionException e)
76     {
77         e.PrintStackTrace();
78         return "duplicate";
79     }
80     catch (FirebaseAuthInvalidUserException e)
81     {
82         e.PrintStackTrace();
83         return string.Empty;
84     }
85     catch (FirebaseAuthInvalidCredentialsException e)
86     {
87         e.PrintStackTrace();
88         return string.Empty;
89     }
90 }
91 /** This function gets the currently signed in users UID.
92 */
93 public string GetUid()
94 {
95     var user = FirebaseAuth.Instance.CurrentUser;
96     if (user != null)
97     {
98         try
99         {
100             var uid = user.Uid;
101             return uid;
102         }
103         catch (FirebaseAuthInvalidUserException e)
104         {
105             e.PrintStackTrace();
106             return string.Empty;
107         }
108         catch (FirebaseAuthInvalidCredentialsException e)
109         {
110             e.PrintStackTrace();
111             return string.Empty;
112         }
113     }
114     else
115     {
116         return "";
117     }
118 }
```

```
119 | /** This function sends the password reset email.  
120 | */  
121 | public async Task ResetPassword(string email)  
122 | {  
123 |     await FirebaseAuth.Instance.SendPasswordResetEmailAsync(email);  
124 | }  
125 | }  
126 | }
```



```
1 {
2   "project_info": {
3     "project_number": "637452254914",
4     "firebase_url": "https://application-green-quake-default-rtdb.firebaseio.com",
5     "project_id": "application-green-quake",
6     "storage_bucket": "application-green-quake.appspot.com"
7   },
8   "client": [
9     {
10      "client_info": {
11        "mobilesdk_app_id": "1:637452254914:android:bc2c95ec69db3a6528455b",
12        "android_client_info": {
13          "package_name": "com.ApplicationGreenQuake"
14        }
15      },
16      "oauth_client": [
17        {
18          "client_id": "637452254914-
19 4t61he1vvfrkq3l2d2c65410noqs06b1.apps.googleusercontent.com",
20          "client_type": 3
21        }
22      ],
23      "api_key": [
24        {
25          "current_key": "AIzaSyBm_gRAhnZ6wVQ_j_sn9CNJ6J1aUq8toFw"
26        }
27      ],
28      "services": {
29        "appinvite_service": {
30          "other_platform_oauth_client": [
31            {
32              "client_id": "637452254914-
33 4t61he1vvfrkq3l2d2c65410noqs06b1.apps.googleusercontent.com",
34              "client_type": 3
35            }
36          ]
37        }
38      ],
39      "configuration_version": "1"
40    }
41  }
```

```

1  /*! \class The MainActivity Native Android Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the MainActivity Native Android Class. It had to be modified to make
   certain Nuget Packages and APIs work
7  *
8  */
9  using Acr.UserDialogs;
10 using Android.App;
11 using Android.Content.PM;
12 using Android.Runtime;
13 using Android.OS;
14 using Firebase;
15 using Plugin.CurrentActivity;
16
17 namespace Application_Green_Quake.Droid
18 {
19     [Activity(Label = "Application_Green_Quake", Icon = "@drawable/AppLogo", Theme =
   "@style/MainTheme", MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize |
   ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout |
   ConfigChanges.SmallestScreenSize )]
20     public class MainActivity :
   global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
21     {
22         /** This function makes it that when outside of a pop up is touched the popup
   closes.
23         */
24         public override void OnBackPressed()
25         {
26             Rg.Plugins.Popup.Popup.SendBackPressed(base.OnBackPressed);
27         }
28         protected override void OnCreate(Bundle savedInstanceState)
29         {
30             TabLayoutResource = Resource.Layout.Tabbar;
31             ToolbarResource = Resource.Layout.Toolbar;
32
33             base.OnCreate(savedInstanceState);
34
35             UserDialogs.Init(this);
36             Rg.Plugins.Popup.Popup.Init(this);
37             CrossCurrentActivity.Current.Init(this, savedInstanceState);
38             FirebaseApp.InitializeApp(Application.Context);
39             Xamarin.Essentials.Platform.Init(this, savedInstanceState);
40             Xamarin.Forms.Forms.Init(this, savedInstanceState);
41             Xamarin.Forms.GoogleMaps.Init(this, savedInstanceState); // initialize for
   Xamarin.Forms.GoogleMaps
42             LoadApplication(new App());
43         }
44
45         public override void OnRequestPermissionsResult(int requestCode, string[]
   permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
46         {
47             Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions,
   grantResults);
48             base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
49         }
50     }
51 }

```

```
1  /*! \class The MainApplication Native Android Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
   c00228956@itcarlow.ie
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the MainApplication Native Android Class. It had to be modified to
   make the APKs work
7  *
8  */
9  #if DEBUG
10 using System;
11 using Android.App;
12 using Android.Runtime;
13 using Application_Green_Quake.Models;
14 using Plugin.CurrentActivity;
15
16 [Application(Debuggable = true)]
17 #else
18 using Android.App;
19 using Android.Runtime;
20 using Application_Green_Quake.Models;
21 using Plugin.CurrentActivity;
22 using System;
23
24 [Application(Debuggable = false)]
25 #endif
26 [MetaData("com.google.android.maps.v2.API_KEY",
27           Value = AppConstants.googleMapsApiKey)]
28 public class MainApplication : Application
29 {
30     public MainApplication(IntPtr handle, JniHandleOwnership transer)
31         : base(handle, transer)
32     {
33     }
34
35     public override void OnCreate()
36     {
37         base.OnCreate();
38         CrossCurrentActivity.Current.Init(this);
39     }
40 }
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1"
  android:versionName="1.0" package="com.ApplicationGreenQuake"
  android:installLocation="preferExternal">
3     <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="30" />
4     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
5     <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
6     <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
7     <uses-permission android:name="android.permission.INTERNET" />
8     <uses-permission android:name="android.permission.CAMERA" />
9     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
10    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
11    <application android:label="Application_Green_Quake.Android"
  android:theme="@style/MainTheme">
12        <uses-library android:name="org.apache.http.legacy" android:required="false" />
13        <provider android:name="android.support.v4.content.FileProvider"
  android:authorities="com.ApplicationGreenQuake.fileprovider" android:exported="false"
  android:grantUriPermissions="true">
14            <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
  android:resource="@xml/file_paths"></meta-data>
15        </provider>
16    </application>
17 </manifest>
```