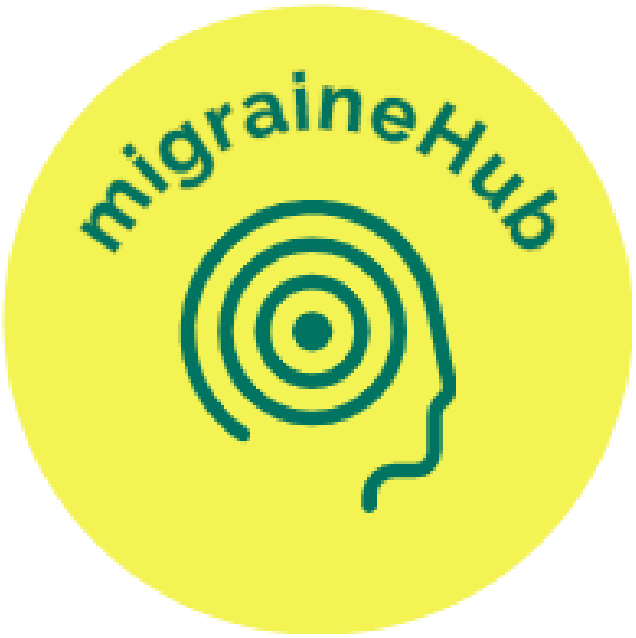
INSTITUTE of TECHNOLOGY CARLOW
Institiúid Teicneolaíochta Cheatharlach



Technical Manual

Student Name: Michelle Bolger

Student Number: C00242743

Supervisor: Dr. Chris Staff

Submission Date: 25/4/2022

Abstract

The inspiration behind this project was to discover more information about migraines, their triggers and symptoms and also to discover information about tracking migraines and the benefits of keeping a journal of migraines.

The purpose of the “migraineHub” app is to develop a cross platform mobile application that will allow people who suffer from migraines to record their migraines (the frequency, type of migraine, pain scale, medications, triggers and symptoms).

Users will also be able to view statistic about their past migraines and email this information to themselves for future reference or for review by their doctor/neurologist.

This technical manual includes the technical details of the migraineHub application. These details include installation instructions and the code developed for this project.

Table of Contents

1. Introduction

This purpose of this manual is to provide installation instructions for the migraineHub application on an Android device.The structure of the code of the project is also discussed. Finally all of the project code is provided.

2. Installation Instructions

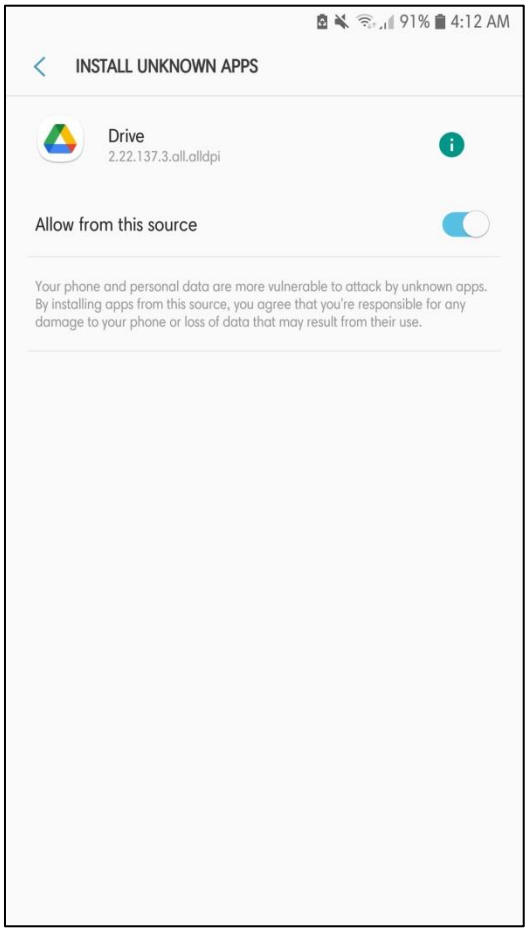


Figure 1 - Allow Install From Drive

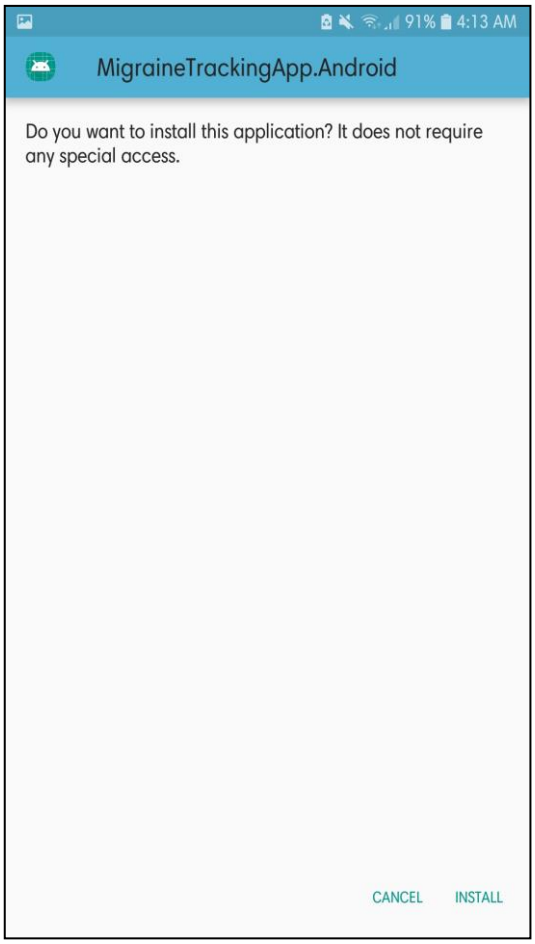


Figure 2 – Install Application

To install the application on an Android device:

1. Download the APK, available here: https://drive.google.com/file/d/1RV0wk9R-5H3FOBTqzFOa3nH257goFUqr/view?usp=sharing
2. As shown in *figure 1,* allowing the installation of applications from Google Drive may be required. Once installation is allowed from this source the APK will download.
3. As shown in *figure 2,* the user can then select “install” and the application will begin installing.

3. Code Structure

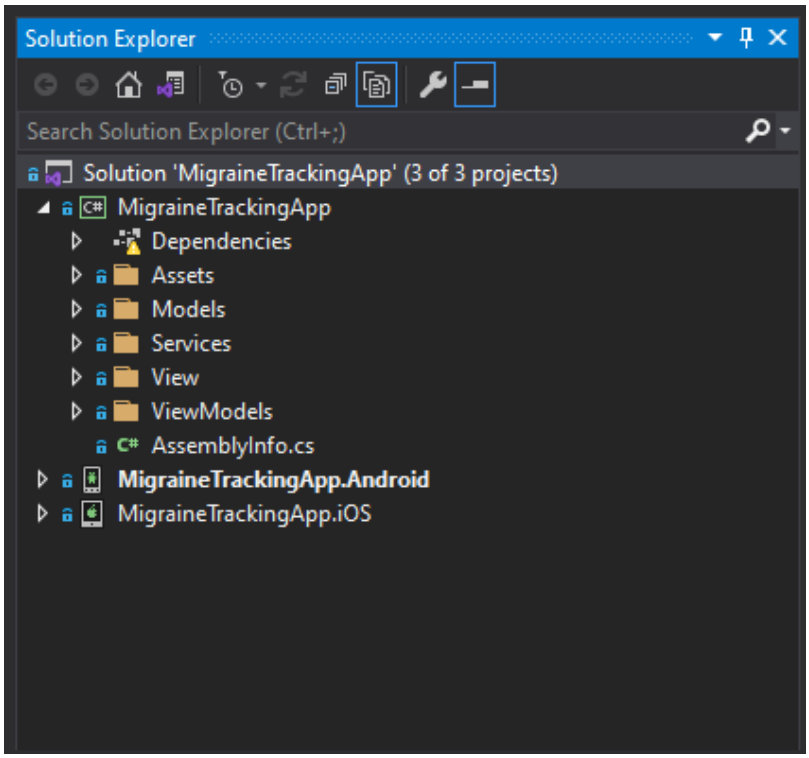


Figure 3 - Code Structure

As shown in *figure 3* the project follows the Model-View-ViewModel (MVVM) design pattern. The Models folder contains objects related to both the database and the APIs used. The Views folder contains all the code for the screens the user/member will see on their device. The ViewModels folder contains the logic between Views and Models. The Services folder contains classes to connect to outside APIs and the database.

4. Code

4.1 Models

4.1.1 Allergen.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 * Date: 18/04/2022
 */

using Newtonsoft.Json;
using System.Collections.Generic;


namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// Sets or gets list of allergens from database
    /// </summary>
    public class Allergen
    {
        [JsonProperty("allergenList")]
        public List<string> allergenList { get; set; }
    }
}
```

4.1.2 Clouds.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    public class Clouds
    {
        [JsonProperty("all")]
        public long All { get; set; }
    }
}
```

4.1.3 Coord.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    public class Coord
    {
        [JsonProperty("lon")]
        public double Lon { get; set; }

        [JsonProperty("lat")]
        public double Lat { get; set; }
    }
}
```

4.1.4 DisplayGraph.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// this object is used for storing values to build microchart graph
    /// </summary>
    public class DisplayGraph
    {
        public string Date { get; set; }

        public float PainLevel { get; set; }
    }
}
```

4.1.5 FoodName.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// returns JSON from openfoodfacts
    /// </summary>
    public class FoodName
    {
        [JsonProperty("product_name")] //finds product name in JSON object
        public string ProductName { get; set; }
        [JsonProperty("allergens_from_ingredients")] //finds allergens from
ingredients in JSON object
        public string Allergens { get; set; }
    }
}
```

4.1.6 Main.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// used in setting variables returned from weather API
    /// </summary>
    public class Main
    {
        [JsonProperty("temp")]
        public double Temperature { get; set; }

        [JsonProperty("pressure")]
        public long Pressure { get; set; }

        [JsonProperty("humidity")]
        public long Humidity { get; set; }

        [JsonProperty("temp_min")]
        public double TempMin { get; set; }

        [JsonProperty("temp_max")]
        public double TempMax { get; set; }
    }
}
```

4.1.7 Member.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using Newtonsoft.Json;
using System;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// gets and sets user profile data from database
    /// </summary>
    public class Member
    {
        [JsonProperty("email")]
        public String Email { get; set; } //variable

        [JsonProperty("firstName")]
        public String FirstName { get; set; } //variable

        [JsonProperty("gender")]
        public String Gender { get; set; } //variable

        [JsonProperty("dob")]
        public String Dob { get; set; } //variable
    }
}
```

4.1.8 Migraine.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using Newtonsoft.Json;
using System.Collections.Generic;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// gets and sets users migraine data from database
    /// </summary>
    public class Migraine
    {
        [JsonProperty("migraineType")]
        public List<string> migraineType { get; set; }

        [JsonProperty("painLocation")]
        public List<string> painLocation { get; set; }

        [JsonProperty("medicationType")]
        public List<string> medicationType { get; set; }

        [JsonProperty("symptoms")]
        public List<string> symptoms { get; set; }

        [JsonProperty("triggers")]
        public List<string> triggers { get; set; }
        [JsonProperty("foods")]
        public List<string> foods { get; set; }

        [JsonProperty("startDate")]
        public string startDate { get; set; }

        [JsonProperty("endDate")]
        public string endDate { get; set; }
        [JsonProperty("startTime")]
        public string startTime { get; set; }

        [JsonProperty("endTime")]
        public string endTime { get; set; }

        [JsonProperty("location")]
        public string location { get; set; }
        [JsonProperty("humidity")]
        public string humidity { get; set; }
        [JsonProperty("temperature")]
        public string temperature { get; set; }
        [JsonProperty("migraineDuration")]
        public string migraineDuration { get; set; }
        [JsonProperty("painIntensity")]
        public string painIntensity { get; set; }
        [JsonProperty("dateEntered")]
        public string dateEntered { get; set; }
    }
}
```

4.1.9 Product.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 * Date: 18/4/2022
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// returns JSON from openfoodfacts
    /// </summary>
    public class Product
    {
        [JsonProperty("product")] //key returned from JSON object
        public FoodName foodDetails { get; set; } //value stored here
    }
}
```

4.1.10 Sys.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    public class Sys
    {
        [JsonProperty("type")]
        public long Type { get; set; }

        [JsonProperty("id")]
        public long Id { get; set; }

        [JsonProperty("message")]
        public double Message { get; set; }

        [JsonProperty("country")]
        public string Country { get; set; }

        [JsonProperty("sunrise")]
        public long Sunrise { get; set; }

        [JsonProperty("sunset")]
        public long Sunset { get; set; }
    }
}
```

4.1.11 Weather.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 * Date: 18/4/2022
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// returns JSON objects from weather API
    /// </summary>
    public class Weather
    {
        [JsonProperty("id")]
        public long Id { get; set; }

        [JsonProperty("main")]
        public string Visibility { get; set; }

        [JsonProperty("description")]
        public string Description { get; set; }

        [JsonProperty("icon")]
        public string Icon { get; set; }
    }
}
```

4.1.12 WeatherData.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// returns JSON objects from weather API
    /// </summary>
    public class WeatherData
    {
        [JsonProperty("name")]
        public string Title { get; set; }

        [JsonProperty("coord")]
        public Coord Coord { get; set; }

        [JsonProperty("weather")]
        public Weather[] Weather { get; set; }

        [JsonProperty("base")]
        public string Base { get; set; }

        [JsonProperty("main")]
        public Main Main { get; set; }

        [JsonProperty("visibility")]
        public long Visibility { get; set; }

        [JsonProperty("wind")]
        public Wind Wind { get; set; }

        [JsonProperty("clouds")]
        public Clouds Clouds { get; set; }

        [JsonProperty("dt")]
        public long Dt { get; set; }

        [JsonProperty("sys")]
        public Sys Sys { get; set; }

        [JsonProperty("id")]
        public long Id { get; set; }

        [JsonProperty("cod")]
        public long Cod { get; set; }
    }
}
```

4.1.13 Wind.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/04/2022
 */

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace MigraineTrackingApp.Models
{
    /// <summary>
    /// used in setting variables returned from weather API
    /// </summary>
    public class Wind
    {
        [JsonProperty("speed")]
        public double Speed { get; set; }

        [JsonProperty("deg")]
        public long Deg { get; set; }
    }
}
```

4.2 Services

4.2.1 DbConnect.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Firebase.Database;
using Firebase.Database.Query;
using Firebase.Storage;
using MigraineTrackingApp.Models;

namespace MigraineTrackingApp.Services
{
    /// <summary>
    /// these methods connect the application to the firebase database
    /// </summary>
    class DbConnect
    {

        FirebaseClient firebase;
        private FirebaseStorage firebaseStorage = new
FirebaseStorage("migrainetrackapp.appspot.com");
        private static Member member = new Member();
        private List<Migraine> migraines = new List<Migraine>();
        public DbConnect()
        {
            firebase = new FirebaseClient("https://migrainetrackapp-default-
rtdb.europe-west1.firebasedatabase.app/");
        }
        public async Task<Member> GetMember(string userId)
        {
            try
            {
                return await
firebase.Child("member").Child(userId).OnceSingleAsync<Member>();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                return null;
            }
        }
        /// <summary>
        /// Create member profile add into database
        /// </summary>
        /// <param name="firstName"></param>
        /// <param name="lastName"></param>
        /// <param name="dob"></param>
        /// <param name="gender"></param>
        /// <param name="userid"></param>
        /// <returns>user profile info</returns>
```

```
        public async Task<bool> createProfile(string firstName,string dob,string
gender,string userid)
        {
            await firebase
                .Child("member").Child(userid)
                .PutAsync(new Member()
                {
                    FirstName = firstName,
                    Dob = dob,
                    Gender = gender,
                });
            return true;
        }
        /// <summary>
        /// This method saves a record of a users migraine data
        /// </summary>
        /// <param name="uid">user ID</param>
        /// <param name="migranetypes"></param>
        /// <param name="painlocation"></param>
        /// <param name="medicationtype"></param>
        /// <param name="symm">symptoms</param>
        /// <param name="trig">triggers</param>
        /// <param name="food">food eaten</param>
        /// <param name="loc">location of migraine</param>
        /// <param name="hum">humidity</param>
        /// <param name="temp">temperture</param>
        /// <param name="sTime">start time</param>
        /// <param name="eTime">end time</param>
        /// <param name="sdate">start date</param>
        /// <param name="edate">end date</param>
        /// <param name="migDuration"></param>
        /// <param name="painInten">pain intensity</param>
        /// <param name="todaysDate"></param>
        /// <returns>bool depending if migraine record succefully created</returns>
        public async Task<bool> createMigraineRecord(string uid,List<string>
migranetypes, List<string> painlocation, List<string> medicationtype, List<string>
symm, List<string> trig, List<string> food,string loc,string hum,string temp,string
sTime,string eTime,string sdate,string edate,string migDuration,string
painInten,string todaysDate)
        {
            try
            {
                await firebase
                .Child("RecordMigraine").Child(uid).Child(todaysDate)
                .PutAsync(new Migraine()
                {
                    migraineType = migranetypes,
                    painLocation = painlocation,
                    medicationType = medicationtype,
                    symptoms = symm,
                    triggers = trig,
                    foods = food,
                    startDate = sdate,
                    endDate = edate,
                    startTime = sTime,
                    endTime = eTime,
                    temperature = temp,
                    location = loc,
                    humidity = hum,
                    migraineDuration = migDuration,
                    painIntensity = painInten,
                    dateEntered = todaysDate
                });
                return true;
```

```
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            return false;
        }
    }
    /// <summary>
    /// gets migraine records from database related to user
    /// </summary>
    /// <param name="uid"> User ID</param>
    /// <returns>list of migraine records</returns>
    public async Task<List<Migraine>> getMigraineRecords(string uid)
    {
        try
        {
               migraines = (await firebase
              .Child("RecordMigraine").Child(uid)
              .OnceAsync<Migraine>()).Select(item => new Migraine
              {
                       endDate = item.Object.endDate,
                       startDate = item.Object.startDate,
                       foods = item.Object.foods,
                       humidity = item.Object.humidity,
                       migraineDuration = item.Object.migraineDuration,
                       location = item.Object.location,
                       medicationType = item.Object.medicationType,
                       migraineType = item.Object.migraineType,
                       painIntensity = item.Object.painIntensity,
                       symptoms = item.Object.symptoms,
                       triggers = item.Object.triggers,
                       endTime = item.Object.endTime,
                       painLocation = item.Object.painLocation,
                       startTime = item.Object.startTime,
                       temperature = item.Object.temperature,
                       dateEntered = item.Object.dateEntered

                }).ToList();
            return migraines;
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            return null;
        }
    }
    /// <summary>
    /// This method adds allergens to a list of allergens in the data base
    /// </summary>
    /// <param name="userid"></param>
    /// <param name="migranetypes"></param>
    /// <returns></returns>
    public async Task<bool> addAllergens(string userid,List<string> migranetypes)
    {
        try
        {
            await firebase
            .Child("allergyList").Child(userid)
            .PutAsync(new Allergen()
            {
                allergenList = migranetypes
            });
            return true;
        }
```

```
            catch (Exception e)
            {
                Console.WriteLine(e.StackTrace);
                return false;
            }
        }
        /// <summary>
        /// This method returns a list of allergens
        /// </summary>
        /// <param name="userid">user ID</param>
        /// <returns>list of strings</returns>
        public async Task<List<string>> getAllergenListFromDb(string userid)
        {
            try
            {
               var allergens = (await firebase
                   .Child("allergyList").Child(userid).OnceSingleAsync<Allergen>());
                if(allergens != null)
                {
                    return allergens.allergenList;
                }
                else{
                    return null;
                }

            }
            catch (Exception e)
            {
                Console.WriteLine(e.StackTrace);
                return new List<string>();
            }
        }


    }
}
```

4.2.2 OpenfoodFacts.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using MigraineTrackingApp.Models;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

namespace MigraineTrackingApp.Services
{
    /// <summary>
    /// this class accesses the openfoodfacts API through Http requests
    /// </summary>
    public class OpenFoodFacts
    {
        string url = "https://world.openfoodfacts.org/api/v0/product/";
        HttpClient _client;
        string[] foodDetails = new string[2];

        /// <summary>
        /// this creates instance of http client
        /// </summary>
        public OpenFoodFacts()
        {
            _client = new HttpClient();
        }
        /// <summary>
        /// this method takes in a barcode as a string attaches to the openfoodfacts
url, sends the string in a query to openfoodfacts
        /// gets response if 200 will deserialise JSON response and stores in models
        /// if 400 response fails
        /// </summary>
        /// <param name="barcode"></param>
        /// <returns></returns>
        public async Task<string[]> getFoodNameFromBarcode(string barcode)
        {
            try
            {
                string query = url + barcode;
                var response = await _client.GetAsync(query);
                if (response.IsSuccessStatusCode)
                {
                    var content = await response.Content.ReadAsStringAsync(); //gets
back JSON object from openfoodfacts
                    Product product = JsonConvert.DeserializeObject<Product>(content);
//formats JSON object
                    string productName = product.foodDetails.ProductName; //find
product name from JSON object
                    string allergens = product.foodDetails.Allergens; // find allergen
list from JSON object
                    foodDetails[0] = productName;
                    foodDetails[1] = allergens;
                    query = "";
```

```
                    return foodDetails;
                }
            }
            catch (Exception ex) // if bad response
            {
                foodDetails[0] = "Could Not Scan Item!, Please Try Again Or Enter
Product Manually On Previous Screen";
                return foodDetails;
            }

            //return foodData;
            return null;
        }
    }
}
```

4.3 Views

4.3.1 AllergenScanner.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.AllergenScanner"
             xmlns:zxing="clr-
namespace:ZXing.Net.Mobile.Forms;assembly=ZXing.Net.Mobile.Forms">

    <StackLayout BackgroundColor="#00905E">

        <Frame BackgroundColor="#00905E" Padding="24" CornerRadius="0">
            <Label Text="Allergen Checker - Scan an item to check if the allergens
affect your migraines!" HorizontalTextAlignment="Center" TextColor="White"
FontSize="26"/>
        </Frame>

            <Label x:Name="scanResultText" TextColor="White" FontSize="25"  />

        <Button x:Name="allergin"
                Text="Check Allergens"
                BackgroundColor="#014941"
                Margin="70,20,70,20"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"
                Clicked="checkAllergens"/>



        <Grid VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand">
            <zxing:ZXingScannerView IsScanning="True"
                                    OnScanResult="ZXingScannerView_OnScanResult" />
            <zxing:ZXingDefaultOverlay
            x:Name="scannerOverlay"
            BottomText="Place the red line over the barcode." />
        </Grid>

    </StackLayout>

</ContentPage>
```

4.3.2 AllergenScanner.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Services;
using MigraineTrackingApp.ViewModels;
using System;
using System.Linq;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]

    public partial class AllergenScanner : ContentPage
    {
        OpenFoodFacts food = new OpenFoodFacts();
        RecordMigraneViewModel vm;
        string[] foodDetails = new string[2];
        string id = " ";
        string allergen = " ";

        /// <summary>
        ///
        /// </summary>
        /// <param name="userid"></param>
        public AllergenScanner(string userid)
        {
            InitializeComponent();
            id = userid;
            vm = new RecordMigraneViewModel();
        }
       /// <summary>
       /// scans iten with ZXing barcode scanner
       /// gets barcode from item sends to openfoodfacts
       /// </summary>
       /// <param name="result"></param>
        private void ZXingScannerView_OnScanResult(ZXing.Result result)
        {
            Device.BeginInvokeOnMainThread(async () =>
            {
                foodDetails = await food.getFoodNameFromBarcode(result.Text);
                bool isTrue = foodDetails[0].Equals("Could Not Scan Item!");
                if (isTrue == false)
                {
                    scanResultText.Text = "Item Scanned!";
                    allergen = foodDetails[1];
                }
                else{
                    scanResultText.Text = foodDetails[0];
                }
            });
        }
        /// <summary>
        /// button when clicked checks allergens from openfoodfacts against the
allergens stored in database for the user
```

```
        /// </summary>
        /// <param name="sender"> </param>
        /// <param name="e"></param>
        private async void checkAllergens(object sender, EventArgs e)
        {
            string allergenResult = "";
            if (allergen != "")
            {
                string newAllergen = allergen.Substring(3);
                string[] allergens = newAllergen.Split(',');
                allergenResult = await vm.checkScannedItem(id, allergens);
                await DisplayAlert("Allergen Alert!", allergenResult, "OK");

               scanResultText.Text = allergenResult;
                allergen = " ";
            }
            else
            {
                await DisplayAlert("Alert!","Did Not Find Allergens, Check
openfoodfacts.org For More Information", "OK");
            }
        }
    }
}
```

4.3.3 App.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.App">
    <Application.Resources>

    </Application.Resources>
</Application>
```

4.3.4 App.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp
{
    public partial class App : Application
    {
        public App()
        {

            InitializeComponent();

            MainPage = new NavigationPage(new MainPage());
        }

        protected override void OnStart()
        {
        }

        protected override void OnSleep()
        {
        }

        protected override void OnResume()
        {
        }
    }
}
```

4.3.5 BarcodeScanner.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.BarcodeScanner"
             xmlns:zxing="clr-
namespace:ZXing.Net.Mobile.Forms;assembly=ZXing.Net.Mobile.Forms">

    <StackLayout BackgroundColor="#00905E">

        <Frame BackgroundColor="#00905E" Padding="24" CornerRadius="0">
            <Label Text="Barcode Scanner!" HorizontalTextAlignment="Center"
TextColor="White" FontSize="26"/>
        </Frame>

        <Label x:Name="scanResultText" TextColor="White" FontSize="25"/>


        <Button
                BackgroundColor="#68C551"
                Grid.Row="5"
                Grid.Column="0"
                Text="Add to FOOD List"
                TextColor="Black"
                CornerRadius="10"
                Margin="70,5,70,5"
                Clicked="returnPrevious"/>



        <Grid VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand">


            <zxing:ZXingScannerView IsScanning="True"
                                    OnScanResult="ZXingScannerView_OnScanResult" />



            <zxing:ZXingDefaultOverlay
            x:Name="scannerOverlay"
            BottomText="Place the red line over the barcode." />


        </Grid>



    </StackLayout>

</ContentPage>
```

4.3.6 BarcodeScanner.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Services;
using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class BarcodeScanner : ContentPage
    {
        OpenFoodFacts food = new OpenFoodFacts();
        RecordMigraneViewModel vm;
        List<string> foodNames = new List<string>();
        string[] foodDetails = new string[2];
        internal BarcodeScanner(RecordMigraneViewModel migraneVM)
        {
            InitializeComponent();
            vm = migraneVM;
        }
        /// <summary>
        /// scans iten with ZXing barcode scanner
        /// gets barcode from item sends to openfoodfacts
        /// </summary>
        /// <param name="result"></param>
        private void ZXingScannerView_OnScanResult(ZXing.Result result)
        {
            Device.BeginInvokeOnMainThread(async () =>
            {
                foodDetails = await food.getFoodNameFromBarcode(result.Text);
                bool isTrue = foodDetails[0].Equals("Could Not Scan Item!, Please Try
Again Or Enter Product Manually On Previous Screen");
                if (isTrue == false)
                {
                    if (foodNames.Contains(foodDetails[0]))
                    {

                    }
                    else
                    {
                        foodNames.Add(foodDetails[0]);
                    }
                }
                scanResultText.Text = foodDetails[0];
            });
        }
       /// <summary>
       /// button saves allergens that have benn scanned
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="args"></param>
        private async void returnPrevious(object sender, EventArgs args)
```

```
        {
            vm.setFoodEaten(foodNames);
            string allergen = foodDetails[1];
            if(allergen != "")
            {
                string newAllergen = allergen.Substring(3);
                string[] allergens = newAllergen.Split(',');
                vm.setAllAllergenypes(allergens);
            }
            await Navigation.PopModalAsync();
        }
    }
}
```

4.3.7 CreateAccountPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.CreateAccountPage"
             xmlns:magic="clr-namespace:MagicGradients;assembly=MagicGradients"
>

    <Grid>

        <magic:GradientView VerticalOptions="FillAndExpand">
            <magic:GradientView.GradientSource>
                <magic:LinearGradient Angle="180">
                    <magic:GradientStop Color="#007361" Offset="0" />
                    <magic:GradientStop Color="#F3F354" Offset="1" />
                </magic:LinearGradient>
            </magic:GradientView.GradientSource>
        </magic:GradientView>



        <StackLayout Padding="0,50,0,0">



            <Image x:Name="logoImage"
               Aspect="AspectFit"
               HorizontalOptions="FillAndExpand"
               HeightRequest="250"
               Margin="20,0,30,10"/>

            <Entry x:Name="memberEmail"
               Placeholder="Email"
               PlaceholderColor="#014941"
               TextColor="#014941"
               Keyboard="Email"
               VerticalOptions="Center"/>

            <Entry x:Name="createAccPassWord"
               Placeholder="Password"
               PlaceholderColor="#014941"
               TextColor="#014941"
               IsPassword="True"
               VerticalOptions="Center"/>


            <Entry x:Name="confirmAccPassWord"
               Placeholder="Confirm Password"
               PlaceholderColor="#014941"
               TextColor="#014941"
               IsPassword="True"
               VerticalOptions="Center"/>

            <Button x:Name="createAccButton"
                Text="Create Account"
                BackgroundColor="#014941"
                Margin="70,20,70,20"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"
```

```
                Clicked="createAccount"/>

        </StackLayout>
    </Grid>

</ContentPage>
```

4.3.8 CreateAccountPage.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CreateAccountPage : ContentPage
    {
        IAuth auth;
        public CreateAccountPage(IAuth auth)
        {
            InitializeComponent();
            var assemble = typeof(CreateAccountPage);
            logoImage.Source =
ImageSource.FromResource("MigraineTrackingApp.Assets.Images.logo.png", assemble);
            this.auth = auth;
        }
        private async void createAccount(object sender, EventArgs e)
        {
            string uid  = await auth.SignupWithEmailPassword(memberEmail.Text,
confirmAccPassWord.Text);
            await Navigation.PushModalAsync(new MainFeedPage(uid,
memberEmail.Text,auth));
        }
    }
}
```

4.3.9 DisplayPainIntensityChart.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.DisplayPainIntensityChart"
             xmlns:chart="clr-namespace:Microcharts.Forms;assembly=Microcharts.Forms"
             BackgroundColor="#005248">
    <ScrollView>
        <StackLayout>
            <Label Text = "Pain Level Chart" TextColor ="white" FontAttributes="Bold"
HorizontalTextAlignment="Center"/>
            <Frame BorderColor="#005248">
                <chart:ChartView x:Name="MyLineChart" HeightRequest="505"  />

            </Frame>
            <Label TextColor="White" FontSize="Medium" FontFamily="Arial"
HorizontalTextAlignment="Center" Text="Medication Taken During The Month"/>

            <ListView  x:Name="showListView" ItemsSource="{Binding meds}">
                <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                            <Label
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Start"
                                   Text="{Binding .}"/>
                        </ViewCell>
                </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>
</ContentPage>
```

4.3.10 DisplayPainIntensityChart.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using Microcharts;
using MigraineTrackingApp.Models;
using SkiaSharp;
using Syncfusion.SfChart.XForms;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class DisplayPainIntensityChart : ContentPage
    {
        List<DisplayGraph> records;
        private List<Microcharts.ChartEntry> chartEntries = new
List<Microcharts.ChartEntry>();
        Microcharts.ChartEntry chart;
       /// <summary>
       /// populates listview for medication taken for the month
       /// </summary>
       /// <param name="allRecords">list of pain intensity and date/time</param>
       /// <param name="meds">string list of medication taken</param>
        public DisplayPainIntensityChart(List<DisplayGraph> allRecords,List<string>
meds)
        {
            InitializeComponent();
            records = allRecords;
            populateChart();
            MyLineChart.Chart = new LineChart { Entries = chartEntries };
            showListView.ItemsSource = meds;
        }

        /// <summary>
        /// populates microchart with pain intensity value and date/time
        /// </summary>
        private void populateChart()
        {
            foreach (DisplayGraph value in records)
            {
                chart = new Microcharts.ChartEntry(value.PainLevel)
                {
                    Label = value.Date,
                    ValueLabel = value.PainLevel.ToString(),
                    Color = SKColor.Parse("#00AB58")
                };
                chartEntries.Add(chart);
            }
        }
    }
}
```

4.3.11 MainFeedPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.MainFeedPage"
                      xmlns:magic="clr-
namespace:MagicGradients;assembly=MagicGradients"
             NavigationPage.HasNavigationBar="false"
>

    <Grid>

        <magic:GradientView VerticalOptions="FillAndExpand">
            <magic:GradientView.GradientSource>
                <magic:LinearGradient Angle="180">
                    <magic:GradientStop Color="#007361" Offset="0" />
                    <magic:GradientStop Color="#F3F354" Offset="1" />
                </magic:LinearGradient>
            </magic:GradientView.GradientSource>
        </magic:GradientView>


        <ScrollView VerticalScrollBarVisibility="Always">
        <StackLayout Padding="0,20,0,0">

            <Image x:Name="logoImage"
               Aspect="AspectFit"
               HorizontalOptions="FillAndExpand"
               HeightRequest="250"
               Margin="20,0,30,50"
               />



                <Button x:Name="profileButton"
                Text="PROFILE PAGE"
                BackgroundColor="#014941"
                Margin="70,20,70,20"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"
                Clicked="profileButton_Clicked"
                 />




                <Button x:Name="statsButton"
                Text="Allergen Checker"
                BackgroundColor="#014941"
                Margin="70,20,70,20"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"
                Clicked="allergenButton_Clicked"/>

            <Button x:Name="recordMigraine"
                Text="Record Migraine Attack"
                BackgroundColor="#014941"
```

```
                Margin="70,20,70,20"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"

                Clicked="recordMigraineButton_Clicked"/>


            <Button x:Name="prevMigraineRecords"
                Text=" Migraine Records"
                BackgroundColor="#014941"
                Margin="70,20,70,20"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"
                Clicked="recordsButton_Clicked"
               />

            <Button x:Name="showMigraineStats"
                Text=" Migraine Stats"
                BackgroundColor="#014941"
                Margin="70,20,70,20"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"
                Clicked="statsButton_Clicked"
                   />

            <Button x:Name="logout"
                Text="Log out"
                BackgroundColor="#014941"
                Margin="70,70,70,70"
                TextColor="white"
                VerticalOptions="Center"
                    HorizontalOptions="Center"
                CornerRadius="10"
                Clicked="logout_Clicked"
                   />



            </StackLayout>
        </ScrollView>
    </Grid>

</ContentPage>
```

4.3.12 MainFeedPage.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Models;
using MigraineTrackingApp.View;
using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainFeedPage : ContentPage
    {
        string userId = "";
        private ShowMigraineRecordsViewModel vm = new ShowMigraineRecordsViewModel();
        List<Migraine> allRecords;
        List<string> months = new List<string>();
        IAuth auth;

        /// <summary>
        ///
        /// </summary>
        /// <param name="userId"></param>
        /// <param name="email"></param>
        /// <param name="auth"></param>
        public MainFeedPage(string userId,string email, IAuth auth)
        {
            InitializeComponent();
            var assemble = typeof(MainFeedPage);

            this.userId = userId;
            vm.Email = email;
            this.auth = auth;
            logoImage.Source =
ImageSource.FromResource("MigraineTrackingApp.Assets.Images.logo.png", assemble);

        }
        protected async override void OnAppearing()
        {
            base.OnAppearing();
        }
        /// <summary>
        /// goes to record migraine screen
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private async void recordMigraineButton_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushModalAsync(new RecordMigraine(userId,vm.Email,auth));
        }
        /// <summary>
        /// goes to profile screen
```

```
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private async void profileButton_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushModalAsync(new ProfilePage(userId));
        }
        /// <summary>
        /// goes to allergen checker screen
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private async void allergenButton_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushModalAsync(new AllergenScanner(userId));
        }
       /// <summary>
       /// goes to migraine records screen
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="e"></param>
        private async void recordsButton_Clicked(object sender, EventArgs e)
        {
            allRecords = await vm.getAllPrevousMigraineRecords(userId);
            await Navigation.PushModalAsync(new showPreviousRecords(userId, vm.Email,
allRecords,auth));
        }
        /// <summary>
        /// goes to migraine stats screen
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private async void statsButton_Clicked(object sender, EventArgs e)
        {
            allRecords = await vm.getAllPrevousMigraineRecords(userId);
            months = vm.getListOfMonths(allRecords, months);
            await Navigation.PushModalAsync(new SelectMonth(allRecords,months));
        }
       /// <summary>
       /// log out button - logs user put of the app
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="e"></param>
        private async void logout_Clicked(object sender, EventArgs e)
        {
            userId = "";
            auth.Logout();
            await Navigation.PushModalAsync(new MainPage());
        }
        /// <summary>
        /// stops back button from being used - prevents user going back to login
screen after succesfull login
        /// </summary>
        /// <returns></returns>
        protected override bool OnBackButtonPressed() => true;
    }
}
```

4.3.13 MainPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.MainPage"
             xmlns:magic="clr-namespace:MagicGradients;assembly=MagicGradients"
>
    <Grid>

    <magic:GradientView VerticalOptions="FillAndExpand">
        <magic:GradientView.GradientSource>
            <magic:LinearGradient Angle="180">
                    <magic:GradientStop Color="#007361" Offset="0" />
                    <magic:GradientStop Color="#F3F354" Offset="1" />
            </magic:LinearGradient>
        </magic:GradientView.GradientSource>
    </magic:GradientView>




        <StackLayout Padding="0,50,0,0">
        <Image x:Name="logoImage"
               Aspect="AspectFit"

               HorizontalOptions="FillAndExpand"
               HeightRequest="250"
               Margin="20,0,30,50"
               />

        <Entry x:Name="loginEmail"
               Placeholder="Email"
               PlaceholderColor="#014941"
               TextColor="#014941"
               Keyboard="Email"
               VerticalOptions="Center"/>

        <Entry x:Name="loginPassWord"
               Placeholder="Password"
               PlaceholderColor="#014941"
               TextColor="#014941"
               IsPassword="True"
               VerticalOptions="Center"/>


        <Button x:Name="loginButton"
                Text="LOGIN"
                BackgroundColor="#014941"
                Margin="70,20,70,20"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"

                Clicked="loginButton_Clicked"/>



        <Button x:Name="createAccountButton"
```

```
                Text="CREATE ACCOUNT"
                BackgroundColor="#014941"
                Margin="70,20,70,70"
                TextColor="white"
                VerticalOptions="Center"
                CornerRadius="10"

                Clicked="createAccountButton_Clicked"/>


    </StackLayout>
    </Grid>

</ContentPage>
```

4.3.14 MainPage.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace MigraineTrackingApp
{
    public partial class MainPage : ContentPage
    {
        IAuth auth; // put in create user too
        bool isEmailEmpty;
        bool isPassEmpty;
        string userID = "";
        public MainPage()
        {
            InitializeComponent();
            NavigationPage.SetHasNavigationBar(this, false);


            var assemble = typeof(MainPage);


            auth = DependencyService.Get<IAuth>(); //
            logoImage.Source =
ImageSource.FromResource("MigraineTrackingApp.Assets.Images.logo.png", assemble);

        }

        /// <summary>
        /// logs in user with successful email and password combo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private async void loginButton_Clicked(object sender, EventArgs e)

        {
            //Reference: https://www.lindseybroos.be/2020/03/xamarin-forms-and-
firebase-authentication/

            isEmailEmpty = string.IsNullOrEmpty(loginEmail.Text); //check if email
input empty or null
            isPassEmpty = string.IsNullOrEmpty(loginPassWord.Text); //check if
password field is empty
            if(!isEmailEmpty && !isPassEmpty)
            {
                userID = await auth.LoginWithEmailPassword(loginEmail.Text,
loginPassWord.Text); //put in create user page
            }

            if (userID != "")
```

```
            {
                await Navigation.PushModalAsync(new MainFeedPage(userID,
loginEmail.Text,auth));
            }
            else
            {
                await DisplayAlert("error", "Invalid Username or Password", "OK");
            }
        }

        /// <summary>
        /// goes to create account page
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>

        private void createAccountButton_Clicked(object sender, EventArgs e)
        {
            Navigation.PushModalAsync(new CreateAccountPage(auth));
        }
        protected override bool OnBackButtonPressed() => true;
    }
}
```

4.3.15 ProfilePage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.ProfilePage"
              xmlns:magic="clr-namespace:MagicGradients;assembly=MagicGradients"
>

    <Grid>

        <magic:GradientView VerticalOptions="FillAndExpand">
            <magic:GradientView.GradientSource>
                <magic:LinearGradient Angle="180">
                    <magic:GradientStop Color="#007361" Offset="0" />
                    <magic:GradientStop Color="#F3F354" Offset="1" />
                </magic:LinearGradient>
            </magic:GradientView.GradientSource>
        </magic:GradientView>

        <ScrollView VerticalScrollBarVisibility="Always">

            <StackLayout Padding="0,50,0,0">

            <Image x:Name="logoImage"
               Aspect="AspectFit"
               HorizontalOptions="FillAndExpand"
               HeightRequest="250"
               Margin="20,0,10,50"/>

            <Image x:Name="resultImage" HeightRequest="50" WidthRequest="50"/>



            <Label Text="First Name" FontSize="Default" TextColor="white"
FontAttributes="Bold"/>

                <Entry x:Name="memberFirstName"
               Placeholder="First Name"
               PlaceholderColor="#014941"
               TextColor="#014941"
               VerticalOptions="Center"
               IsReadOnly="True"/>

                <Label Text= "Gender" FontSize="Default" TextColor="white"
FontAttributes="Bold"/>


            <Entry x:Name="gen"
               Placeholder="Gender"
               PlaceholderColor="#014941"
               TextColor="#014941"
               VerticalOptions="Center"
               IsReadOnly="True"/>

                <Label Text="DOB" FontSize="Default" TextColor="white"
FontAttributes="Bold"/>

               <Entry x:Name="memberDob"
               Placeholder="Date of Birth"
               PlaceholderColor="#014941"
```

```
                TextColor="#014941"
                VerticalOptions="Center"
                IsReadOnly="True"/>


            <Button Text="Update Profile"
                    Clicked="updateInfo"
                    BackgroundColor="#014941"
                     VerticalOptions="Center"
                     CornerRadius="10"
                    TextColor="white"
                    Margin="70,20,70,20"
                    />

        </StackLayout>
        </ScrollView>
    </Grid>

</ContentPage>
```

4.3.16 ProfilePage.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using MigraineTrackingApp.View;
using MigraineTrackingApp.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ProfilePage : ContentPage
    {
        MemberViewModel memberVm;
        string Id = "";
        private RadioButton button;

        public ProfilePage(string userId)
        {
            InitializeComponent();
            this.Id = userId;

            var assemble = typeof(ProfilePage);

            logoImage.Source =
ImageSource.FromResource("MigraineTrackingApp.Assets.Images.logo.png", assemble);
        }

        protected async override void OnAppearing()
        {

            base.OnAppearing();
            memberVm = new MemberViewModel();
            var member = await memberVm.getMember(Id); //gets list back from viewModel
            if(member != null)
            {
                memberFirstName.Text = member.FirstName;
                gen.Text = member.Gender;
                memberDob.Text = member.Dob;
            }
        }
        /// <summary>
        /// This button goes to the update page
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        async void updateInfo(System.Object sender, System.EventArgs e)
        {
            if(memberFirstName.Text == null)
            {
                memberFirstName.Text = " ";
            }
            if(memberDob.Text == null)
            {
                memberDob.Text = " ";
            }
            if (gen.Text == null)
```

```
            {
                gen.Text = " ";
            }
            await Navigation.PushModalAsync(new UpdateProfile(memberFirstName.Text,
memberDob.Text, gen.Text, Id));
        }
    }
}
```

4.3.17 RecordDate.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordDate" BackgroundColor="#005248">

    <StackLayout Margin="10">
        <Label Text="Choose Start and End Date of Migraine Attack"
               Style="{DynamicResource TitleStyle}"
               Margin="0, 20"
               TextColor="White"
               HorizontalTextAlignment="Center" />

        <Label Text="Start Date:"
               TextColor="White"/>

        <DatePicker x:Name="startDatePicker"
                    Format="D"
                    Margin="30, 0, 0, 30"
                    DateSelected="OnDateSelected"
                    TextColor="White"/>

        <Label Text="End Date:"
               TextColor="White"/>

        <DatePicker x:Name="endDatePicker"
                    MinimumDate="{Binding Source={x:Reference startDatePicker},
                                          Path=Date}"
                    Format="D"
                    Margin="30, 0, 0, 30"
                    DateSelected="OnDateSelected"
                    TextColor="White"/>

        <StackLayout Orientation="Horizontal"
                     Margin="0, 0, 0, 30">
            <Label Text="Record Length Of Migraine Attack? "
                   VerticalOptions="Center"
                   TextColor="White"/>

            <Switch x:Name="includeSwitch"
                    Toggled="OnSwitchToggled"
                    />
        </StackLayout>
        <Label x:Name="resultLabel"
               FontAttributes="Bold"
               HorizontalTextAlignment="Center"
               TextColor="White"/>

        <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="SAVE"
                TextColor="Black"
                Clicked="saveDates"
                CornerRadius="10" />

    </StackLayout>
</ContentPage>
```

4.3.18 RecordDate.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.ViewModels;
using System;
using System.Globalization;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordDate : ContentPage
    {
        RecordMigraneViewModel migraneVM;
        internal RecordDate(RecordMigraneViewModel migraneVM)
        {
            InitializeComponent();
            this.migraneVM = migraneVM;
            if(migraneVM.StartDate != " " && migraneVM.StartDate != null)
            {
                //DateTime dateTime = DateTime.Parse(migraneVM.StartDate,
CultureInfo.CreateSpecificCulture("en-US")); //converts date string from database to
date time format for date picker
                //string newStartDate = dateTime.ToString("dd/MM/yyyy");
                startDatePicker.Date = DateTime.Parse(migraneVM.StartDate);
            }
            if(migraneVM.EndDate != " " && migraneVM.EndDate != null)
            {
                //DateTime dateTime = DateTime.Parse(migraneVM.EndDate,
CultureInfo.CreateSpecificCulture("en-US"));
                //string newEndDate = dateTime.ToString("dd/MM/yyyy");
                endDatePicker.Date = DateTime.Parse(migraneVM.EndDate);
            }
        }

        void OnDateSelected(object sender, DateChangedEventArgs args)
        {
            Recalculate();
        }

        void OnSwitchToggled(object sender, ToggledEventArgs args)
        {
           Recalculate();
        }

        /// <summary>
        /// calculates amount of days inbetween start and end date
        /// </summary>
        void Recalculate()
        {
            //Ref: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-
interface/datepicker

            TimeSpan timeSpan = endDatePicker.Date - startDatePicker.Date +
                (includeSwitch.IsToggled ? TimeSpan.FromDays(1) : TimeSpan.Zero);
```

```
            resultLabel.Text = String.Format("{0} day{1} between dates",
                                                timeSpan.Days, timeSpan.Days == 1 ? ""
: "s");



        }

        /// <summary>
        /// return to menu
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private async void returnToMenu(object sender, EventArgs args)
        {
            await Navigation.PopModalAsync();
        }

       /// <summary>
       /// saves selected start and end dates
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="args"></param>
        private async void saveDates(object sender, EventArgs args)
        {
            string sTime = startDatePicker.Date.ToString();
            string eTime = endDatePicker.Date.ToString();
            int pos = sTime.IndexOf(" ");
            string date  = sTime.Substring(0, pos);
            int ePos = eTime.IndexOf(" ");
            string eDate = eTime.Substring(0, ePos);
            migraneVM.StartDate = date;
            migraneVM.EndDate = eDate;
            await Navigation.PopModalAsync();
        }
    }
}
```

4.3.19 RecordFood.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordFood" BackgroundColor="#005248">

    <StackLayout Margin="10">
        <Label Text="Scan Food Items You Have Eaten Today"
               Style="{DynamicResource TitleStyle}"
               Margin="0, 20"
               TextColor="White"
               HorizontalTextAlignment="Center"/>


        <Button
                BackgroundColor="#00905E"
                Grid.Row="5"
                Grid.Column="0"
                Text="SCAN FOOD ITEM"
                TextColor="Black"
                CornerRadius="10"
                Clicked="goToScan"/>



        <Entry x:Name="addFood"
               Placeholder="Enter Additional Food Items Here"
               PlaceholderColor="White"
               TextColor="White"
               VerticalOptions="Center"/>


        <Button
                BackgroundColor="#00905E"
                Grid.Row="5"
                Grid.Column="0"
                Text="ADD"
                TextColor="Black"
                CornerRadius="10"
                Clicked="addToList"/>

        <Label Text="Food Eaten" HorizontalTextAlignment="Center" FontSize="Medium"/>

        <StackLayout>
            <ListView  HeightRequest="500" x:Name="showListView" ItemsSource="{Binding
selectedTriggers}" Margin="20">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
```

```
                                    HorizontalTextAlignment="Center"
                                    Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Remove"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="removeFromList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>


    <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="SAVE"
                TextColor="Black"
                Clicked="saveFoodList"
                CornerRadius="10" />

    </StackLayout>
</ContentPage>
```

4.3.20 RecordFood.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2021
 */

using MigraineTrackingApp.Services;
using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordFood : ContentPage
    {
        List<string> food = new List<string>();
        RecordMigraneViewModel migraneVM;
        OpenFoodFacts barcode = new OpenFoodFacts();
        internal RecordFood(RecordMigraneViewModel migraneVM)
        {
            InitializeComponent();
            this.migraneVM = migraneVM;
        }
       /// <summary>
       /// populates list view with list of food items
       /// </summary>
        protected async override void OnAppearing()
        {
            base.OnAppearing();
            List<string> scannedItems = migraneVM.getFoodEaten();
            if (food.Count != 0)
            {
                if (scannedItems.Count != 0)
                {
                    food.AddRange(scannedItems);
                    showListView.ItemsSource = null;
                    showListView.ItemsSource = food;
                }
            }
              else
            {
                if (scannedItems.Count != 0)
                {
                    food.AddRange(scannedItems);
                    showListView.ItemsSource = null;
                    showListView.ItemsSource = food;
                }
            }
            migraneVM.resetFoodList();
            if (migraneVM.getFoodEaten().Count != 0 &&
!migraneVM.getFoodEaten().Contains(" "))
            {
                food.AddRange(migraneVM.getFoodEaten());
                showListView.ItemsSource = migraneVM.getFoodEaten();
            }
        }
       /// <summary>
       /// add scan food to list of food items
```

```
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
         private void addToList(object sender, EventArgs args)
         {
             if (addFood.Text != null)
             {
                 food.Add(addFood.Text);
                 addFood.Text = "";
                 showListView.ItemsSource = null;
                 showListView.ItemsSource = food;
             }
         }

         /// <summary>
         /// go to barcode scanner
         /// </summary>
         /// <param name="sender"></param>
         /// <param name="args"></param>
         private async void goToScan(object sender, EventArgs args)
         {
             await Navigation.PushModalAsync(new BarcodeScanner(migraneVM));
         }


         private void removeFromList(object sender, EventArgs args)
         {
             Button button = (Button)sender;
             string value = button.CommandParameter.ToString();

             food.RemoveAll(x => x.StartsWith(value));
             showListView.ItemsSource = null;
             showListView.ItemsSource = food;
         }
         /// <summary>
         /// Returns to previous menu
         /// </summary>
         /// <param name="sender"></param>
         /// <param name="args"></param>
         private async void returnToMenu(object sender, EventArgs args)
         {
             await Navigation.PopModalAsync();
         }
         /// <summary>
         /// This method saves the food list to a view model
         /// </summary>
         /// <param name="sender"></param>
         /// <param name="args"></param>
         private async void saveFoodList(object sender, EventArgs args)
         {
             if (food.Count != 0)
             {
                 migraneVM.getFoodEaten().Clear();
                 migraneVM.setFoodEaten(food);
                 await Navigation.PopModalAsync();
             }
             else
             {
                 await DisplayAlert("Alert", "You Have Not Entered Any Food!", "OK");
             }
         }
     }
}
```

4.3.21 RecordMedication.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordMedication"
BackgroundColor="#005248">


    <StackLayout>
        <StackLayout>
            <ListView x:Name="MedicationTypeListView" ItemsSource="{Binding
medicationTypes}" Margin="20" SeparatorVisibility="Default">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Add"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="addToList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>

        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.50*" />
                <ColumnDefinition Width="0.50*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="40" />
            </Grid.RowDefinitions>

            <Entry x:Name="addMedicationType"
                   Grid.Row="0" Grid.Column="0"
               Placeholder="Add Another"
               PlaceholderColor="White"
               TextColor="White"
               VerticalOptions="Center"/>
```

```
            <Button
              Grid.Row="0"
                 Grid.Column="1"
                BackgroundColor="#00905E"

                Text="ADD"
                TextColor="Black"
                CornerRadius="10"
                Clicked="addToList"/>

        </Grid>

        <Label Text="Medication Taken" HorizontalTextAlignment="Center"
FontSize="Medium"/>

        <StackLayout>
            <ListView  HeightRequest="500" x:Name="showListView" ItemsSource="{Binding
selectedMedicationTypes}" Margin="20">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="
                                       WHITE"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Remove"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="removeFromList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>


        <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="SAVE"
                TextColor="Black"
                Clicked="saveTypes"
                CornerRadius="10" />

    </StackLayout>
</ContentPage>
```

4.3.22 RecordMedication.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordMedication : ContentPage
    {
        List<string> selectedMedicationType = new List<string>();
        string[] medicationType;
        RecordMigraineViewModel migraneVM;
        internal RecordMedication(RecordMigraineViewModel migraneVM)
        {
            InitializeComponent();
            this.migraneVM = migraneVM;
        }
        protected async override void OnAppearing()
        {
            base.OnAppearing();
            medicationType = new[] {
"Paracetemol","Ibuprofen","Frovex","Naproxyn","None"  };
            MedicationTypeListView.ItemsSource = medicationType;
            if (migraneVM.getMedicationTypes().Count != 0 &&
!migraneVM.getMedicationTypes().Contains(" "))
            {
                selectedMedicationType.AddRange(migraneVM.getMedicationTypes());
                showListView.ItemsSource = migraneVM.getMedicationTypes();
            }
        }
        /// <summary>
        /// This method adds to a list of medications
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void addToList(object sender, EventArgs args)
        {
            Button button = (Button)sender;
            if (button.CommandParameter != null)
            {
                string value = button.CommandParameter.ToString();
                if (value != null)
                {
                    selectedMedicationType.Add(value);
                }
            }
            else
            {
```

```
                if (addMedicationType.Text != null)
                {
                    selectedMedicationType.Add(addMedicationType.Text);
                    addMedicationType.Text = "";
                }
            }
            showListView.ItemsSource = null;
            showListView.ItemsSource = selectedMedicationType;

        }
        /// <summary>
        /// This method removes medications from the list
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void removeFromList(object sender, EventArgs args)
        {
            Button button = (Button)sender;
            string value = button.CommandParameter.ToString();

            selectedMedicationType.RemoveAll(x => x.StartsWith(value));
            showListView.ItemsSource = null;
            showListView.ItemsSource = selectedMedicationType;
        }
        /// <summary>
        /// Returns to previous menu
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private async void returnToMenu(object sender, EventArgs args)
        {
            await Navigation.PopModalAsync();
        }
        /// <summary>
        /// This method saves the meds list to a view model
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private async void saveTypes(object sender, EventArgs args)
        {
            if (selectedMedicationType.Count != 0)
            {
                migraneVM.getMedicationTypes().Clear();
                migraneVM.setMedicationTypes(selectedMedicationType);
                await Navigation.PopModalAsync();
            }
            else
            {
                await DisplayAlert("Alert", "You Have Not Selected Any Medication",
"OK");
            }
        }
    }
}
```

4.3.23 RecordMigraine.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.RecordMigraine"
             BackgroundColor="#005248"
              NavigationPage.HasNavigationBar="false">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition  Width="*" />
        </Grid.ColumnDefinitions>

        <Grid RowSpacing="30" ColumnSpacing="30" ></Grid>

        <Button x:Name="recordTimeButton"
                BackgroundColor="#007361"
                Grid.Row="0"
                Grid.Column="0"
                Text="Time"
                TextColor="White"
                CornerRadius="10"

                Clicked="recordTimeButton_Clicked"/>


        <Button x:Name="recordDateButton"
                BackgroundColor="#007361"
                Grid.Row="0"
                Grid.Column="1"
                Text="Date"
                TextColor="White"
                CornerRadius="10"
                Clicked="recordDateButton_Clicked" />


        <Button x:Name="recordTypeButton"
                BackgroundColor="#007361"
                Grid.Row="1"
                Grid.Column="0"
                Text="Migraine/Headache Type"
                TextColor="White"
                CornerRadius="10"
                Clicked="recordTypeButton_Clicked" />



        <Button x:Name="recordPainIntensity"
                BackgroundColor="#007361"
                Grid.Row="1"
                Grid.Column="1"
                Text="Pain Intensity"
                TextColor="White"
                CornerRadius="10"
```

```
        Clicked="recordPainIntensityButton_Clicked"/>


<Button x:Name="recordPainLocationButton"
        BackgroundColor="#007361"
        Grid.Row="2"
        Grid.Column="0"
        Text="Pain Location"
        TextColor="White"
        CornerRadius="10"
        Clicked="recordpainLocationButton_Clicked"/>


<Button x:Name="recordMedTypes"
        BackgroundColor="#007361"
        Grid.Row="2"
        Grid.Column="1"
        Text="Medication"
        TextColor="White"
        CornerRadius="10"
        Clicked="recordMedTypeButton_Clicked" />

<Button x:Name="recordSymptoms"
        BackgroundColor="#007361"
        Grid.Row="3"
        Grid.Column="0"
        Text="Symptoms"
        TextColor="White"
        CornerRadius="10"
        Clicked="recordSymptomsButton_Clicked" />


<Button x:Name="recordTriggers"
        BackgroundColor="#007361"
        Grid.Row="3"
        Grid.Column="1"
        Text="Triggers"
        TextColor="White"
        CornerRadius="10"
        Clicked="recordTriggersButton_Clicked"/>


<Button x:Name="recordFoodButton"
        BackgroundColor="#007361"
        Grid.Row="4"
        Grid.Column="0"
        Text="Food Eaten"
        TextColor="White"
        CornerRadius="10"
        Clicked="recordFoodButton_Clicked"/>

<Button x:Name="recordWeather"
        BackgroundColor="#007361"
        Grid.Row="4"
        Grid.Column="1"
        Text="Weather"
        TextColor="White"
        CornerRadius="10"
        Clicked="recordWeatherButton_Clicked"/>

<Button x:Name="backButton"
        BackgroundColor="#00AB58"
        Grid.Row="5"
        Grid.Column="0"
```

```
                Text="BACK"
                TextColor="Black"
                CornerRadius="10"
                Clicked="backButton_Clicked"/>

        <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="SAVE"
                TextColor="Black"
                CornerRadius="10"
                Clicked="savePlan"/>


    </Grid>

</ContentPage>
```

4.3.24 RecordMigraine.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using MigraineTrackingApp.ViewModels;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordMigraine : ContentPage
    {
        RecordMigraneViewModel migraneVM = new RecordMigraneViewModel();
        DateTime currentDate;
        string id = "";
        string email = "";
        IAuth auth;
        public RecordMigraine(string userId, string email, IAuth auth)
        {
            InitializeComponent();
            DateTime now = DateTime.Now;
            currentDate = now;
            id = userId;
            this.email = email;
            this.auth = auth;
        }
        internal RecordMigraine(string userId,RecordMigraneViewModel vm,string email,
IAuth auth)
        {
            InitializeComponent();
            DateTime now = DateTime.Now; // set date to current date
            currentDate = now;
            id = userId;
            migraneVM = vm;
            this.email = email;
            this.auth = auth;
        }
       /// <summary>
       /// depends on which way record migraine is entered
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="args"></param>
        private async void savePlan(object sender, EventArgs args)
        {
            if(migraneVM.StartDate != " " && migraneVM.StartTimeOfMigraine == " ")
            {
                migraneVM.checkIfAllergensAreInDB(id);
                migraneVM.sendAllergenInfo(id);
                migraneVM.sendRecordDetailsToDataase(migraneVM.StartDate, id);
            }
            else if(migraneVM.StartDate != null && migraneVM.StartTimeOfMigraine !=
null)
            {
                string date = migraneVM.StartDate + " " +
migraneVM.StartTimeOfMigraine;
                string newDate = date.Replace("/", "-");
```

```
                migraneVM.checkIfAllergensAreInDB(id);
                migraneVM.sendAllergenInfo(id);
                migraneVM.sendRecordDetailsToDataase(newDate, id);
            }
            else if (migraneVM.StartDate != null && migraneVM.StartTimeOfMigraine ==
null)
            {
                int firstSpaceIndex = migraneVM.StartDate.IndexOf(" ");//get first
spcae
                string date = migraneVM.StartDate.Substring(0, firstSpaceIndex);
                date = date + " " + migraneVM.StartTimeOfMigraine;
                string newDate = date.Replace("/", "-");
                migraneVM.checkIfAllergensAreInDB(id);
                migraneVM.sendAllergenInfo(id);
                migraneVM.sendRecordDetailsToDataase(newDate, id);
            }
            else
            {
                string newDate = currentDate.ToString("dd'/'MM'/'yyyy HH:mm:ss");
                int firstSpaceIndex = newDate.IndexOf(" ");//get first spcae
                string date = newDate.Substring(0, firstSpaceIndex);
                string time = newDate.Substring(firstSpaceIndex+1);
                newDate = newDate.Replace("/", "-");
                migraneVM.StartDate = date;
                migraneVM.StartTimeOfMigraine = time;
                migraneVM.checkIfAllergensAreInDB(id);
                migraneVM.sendAllergenInfo(id);
                migraneVM.sendRecordDetailsToDataase(newDate, id);
            }
        }

        private async void backButton_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushModalAsync(new MainFeedPage(id, email, auth));
        }

       /// <summary>
       /// go to record date page
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="e"></param>
        private async void recordDateButton_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushModalAsync(new View.RecordDate(migraneVM));
        }

       /// <summary>
       /// go to record type page
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="e"></param>
        private async void recordTypeButton_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushModalAsync(new View.RecordMigraineType(migraneVM));
        }

       /// <summary>
       /// go to record pain location page
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="e"></param>
        private async void recordpainLocationButton_Clicked(object sender, EventArgs
e)
        {
```

```
        await Navigation.PushModalAsync(new View.RecordPainLocation(migraneVM));
    }

    /// <summary>
    /// go to record medication taken page
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private async void recordMedTypeButton_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushModalAsync(new View.RecordMedication(migraneVM));
    }

   /// <summary>
   /// go to record symptom page
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
    private async void recordSymptomsButton_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushModalAsync(new View.RecordSymptoms(migraneVM));
    }

  /// <summary>
  /// go to record triggers page
  /// </summary>
  /// <param name="sender"></param>
  /// <param name="e"></param>
    private async void recordTriggersButton_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushModalAsync(new View.RecordTriggers(migraneVM));
    }

    /// <summary>
    /// go to record weather page
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private async void recordWeatherButton_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushModalAsync(new View.RecordWeather(migraneVM));
    }

  /// <summary>
  /// go to record time page
  /// </summary>
  /// <param name="sender"></param>
  /// <param name="e"></param>
    private async void recordTimeButton_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushModalAsync(new View.RecordTime(migraneVM));
    }

   /// <summary>
   /// go to record food page
   /// </summary>
   /// <param name="sender"></param>
   /// <param name="e"></param>
    private async void recordFoodButton_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushModalAsync(new View.RecordFood(migraneVM));
    }

  /// <summary>
```

```
        /// go to record pain intensity page
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
          private async void recordPainIntensityButton_Clicked(object sender, EventArgs
e)
          {
              await Navigation.PushModalAsync(new View.RecordPainIntensity(migraneVM));
          }
          protected override bool OnBackButtonPressed() => true;


    }
}
```

4.3.25 RecordMigraineType.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordMigraineType"
             BackgroundColor="#005248">


        <StackLayout>
        <StackLayout>
            <ListView x:Name="MigraineTypeListView" ItemsSource="{Binding
migraineTypes}" Margin="10" SeparatorVisibility="Default">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Add"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="addToList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>

        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.50*" />
                <ColumnDefinition Width="0.50*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="40" />
            </Grid.RowDefinitions>

            <Entry x:Name="addMigraineType"
                   Grid.Row="0" Grid.Column="0"
               Placeholder="Add Another"
               PlaceholderColor="White"
               TextColor="White"
               VerticalOptions="Center"/>
```

```
            <Button
                 Grid.Row="0"
                                Grid.Column="1"
                BackgroundColor="#00905E"

                Text="ADD"
                TextColor="Black"
                CornerRadius="10"
                Clicked="addToList"/>

        </Grid>

        <Label Text="Migraine/Headache Type" HorizontalTextAlignment="Center"
FontSize="Medium"/>

        <StackLayout>
            <ListView  HeightRequest="500" x:Name="showListView" ItemsSource="{Binding
selectedMigraneTypes}" Margin="20">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Remove"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="removeFromList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>


            <Button x:Name="saveButton"
                BackgroundColor="#00AB58"

                Text="SAVE"
                    Margin="70,10,70,10"
                TextColor="Black"
                Clicked="saveTypes"
                CornerRadius="10" />

        </StackLayout>
</ContentPage>
```

4.3.26 RecordMigraineType.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordMigraineType : ContentPage
    {
        List<string> selectedMigraneTypes = new List<string>();
        string[] migraineTypes;
        RecordMigraneViewModel migraneVM;
        internal RecordMigraineType(RecordMigraneViewModel migraneVM)
        {
            InitializeComponent();
            this.migraneVM = migraneVM;
        }
        protected async override void OnAppearing()
        {
            base.OnAppearing();
            migraineTypes = new[] { "Migraine", "Tension Headache", "Cluster
Headache", "Headache", "Sinus Headache", "Not Sure" };

            MigraineTypeListView.ItemsSource = migraineTypes;
            if (migraneVM.getMigraneTypes().Count != 0 &&
!migraneVM.getMigraneTypes().Contains(" "))
            {
                selectedMigraneTypes.AddRange(migraneVM.getMigraneTypes());
                showListView.ItemsSource = selectedMigraneTypes;
            }
        }
        /// <summary>
        /// This method adds to a list of migrane types
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void addToList(object sender, EventArgs args)
        {
            Button button = (Button)sender;
            if (button.CommandParameter != null)
            {
                string value = button.CommandParameter.ToString();
                if (value != null)
                {
                    selectedMigraneTypes.Add(value);
                }
            }
            else
            {
                if(addMigraineType.Text != null)
                {
                    selectedMigraneTypes.Add(addMigraineType.Text);
```

```
                addMigraineType.Text = "";
            }
        }
        showListView.ItemsSource = null;
        showListView.ItemsSource = selectedMigraneTypes;

    }
    /// <summary>
    /// This method removes migrane types from the list
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="args"></param>
    private void removeFromList(object sender, EventArgs args)
    {
        Button button = (Button)sender;
        string value = button.CommandParameter.ToString();

        selectedMigraneTypes.RemoveAll(x => x.StartsWith(value));
        showListView.ItemsSource = null;
        showListView.ItemsSource = selectedMigraneTypes;
    }
    /// <summary>
    /// Returns to previous menu
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="args"></param>
    private async void returnToMenu (object sender, EventArgs args)
    {
        await Navigation.PopAsync();
    }
    /// <summary>
    /// This method saves the migrane type list to a view model
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="args"></param>
    private async void saveTypes(object sender, EventArgs args)
    {
        if(selectedMigraneTypes.Count != 0)
        {
            migraneVM.getMigraneTypes().Clear();
            migraneVM.setMigraneTypes(selectedMigraneTypes);
            await Navigation.PopModalAsync();
        }
        else
        {
            await DisplayAlert("Alert", "You Have Not Selected Any Migraine Type",
"OK");
        }
    }
}
}
```

4.3.27 RecordPainIntensity.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordPainIntensity"
BackgroundColor="#005248">
    <ContentPage.Content>
        <StackLayout  Margin="10" >

                <Label Grid.Row="0"
                       Grid.Column="0"
                       Text="Choose intensity of Migraine Attack"
                        Style="{DynamicResource TitleStyle}"
                       Margin="0, 20"
                     TextColor="White"
                     HorizontalTextAlignment="Center"/>



                <Picker Grid.Row="0" Grid.Column="1" x:Name="Intensity" Title="Select
Pain Intensity" TitleColor="White" TextColor="White">
                    <Picker.ItemsSource>
                        <x:Array Type="{x:Type x:String}">
                            <x:String>1: No Pain</x:String>
                            <x:String>2: Mild Pain</x:String>
                            <x:String>3: Uncomfortable Pain</x:String>
                            <x:String>4: Moderate Pain</x:String>
                            <x:String>5: Distracting Pain</x:String>
                            <x:String>6: Distressing Pain</x:String>
                            <x:String>7: Constant Pain</x:String>
                            <x:String>8: Intense Pain</x:String>
                            <x:String>9: Severe Pain</x:String>
                            <x:String>10: Excruciating Pain</x:String>
                        </x:Array>
                    </Picker.ItemsSource>
                </Picker>



            <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="SAVE"
                TextColor="black"
                Clicked="saveIntensity"
                    Margin="70,5,70,5"
                CornerRadius="10" />


    </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

4.3.28 RecordPainIntensity.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.ViewModels;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordPainIntensity : ContentPage
    {
        RecordMigraneViewModel mvm;

        internal RecordPainIntensity(RecordMigraneViewModel migraneVM)
        {
            InitializeComponent();
            mvm = migraneVM;
            if(migraneVM.PainIntensity != " " && migraneVM.PainIntensity != null)
            {
                Intensity.SelectedItem = migraneVM.PainIntensity;
            }
        }

       /// <summary>
       /// go to main menu
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="args"></param>
        private async void returnToMenu(object sender, EventArgs args)
        {
            await Navigation.PopAsync();
        }

        /// <summary>
        /// saves pain intensity
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private async void saveIntensity(object sender, EventArgs args)
        {

            if (Intensity.SelectedIndex != -1)
            {

                mvm.PainIntensity= Intensity.SelectedItem.ToString();
                await Navigation.PopModalAsync();
            }
            else
            {
                await DisplayAlert("Alert", "Please Selecr A Level Of Pain Intensity",
"OK");
            }
        }
    }
}
```

4.3.29 RecordPainLocation.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordPainLocation"
BackgroundColor="#005248">


    <StackLayout>
        <StackLayout>
            <ListView x:Name="PainLocationListView" ItemsSource="{Binding
painLocation}" Margin="20" SeparatorVisibility="Default">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Add"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="addToList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>


        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.50*" />
                <ColumnDefinition Width="0.50*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="40" />
            </Grid.RowDefinitions>

            <Entry x:Name="addPainLocation"
                   Grid.Row="0" Grid.Column="0"
               Placeholder="Add Another"
               PlaceholderColor="White"
               TextColor="White"
               VerticalOptions="Center"/>
```

```
        <Button
                 Grid.Row="0"
                 Grid.Column="1"
                BackgroundColor="#00905E"
                Text="ADD"
                TextColor="Black"
                CornerRadius="10"
                Clicked="addToList"/>


        </Grid>

        <Label Text="Pain Location" HorizontalTextAlignment="Center"
FontSize="Medium"/>

        <StackLayout>
            <ListView  HeightRequest="500" x:Name="showListView" ItemsSource="{Binding
selectedPainLocation}" Margin="20">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="WHITE"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Remove"
                                TextColor="BLACK"
                                CornerRadius="10"
                                Clicked="removeFromList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>


        <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="SAVE"
                TextColor="BLACK"
                Clicked="saveTypes"
                CornerRadius="10" />

    </StackLayout>
</ContentPage>
```

4.3.30 RecordPainLocation.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordPainLocation : ContentPage
    {
        List<string> selectedPainLocation = new List<string>();
        string[] painLocation;
        RecordMigraineViewModel migraneVM;
        internal RecordPainLocation(RecordMigraineViewModel migraneVM)
        {
            InitializeComponent();
            this.migraneVM = migraneVM;
        }
        protected async override void OnAppearing()
        {
            base.OnAppearing();
            painLocation = new[] { "Right Eye", "Left Eye", "Forehead (Right)",
"Forehead (Left)", "Not Sure"};
            PainLocationListView.ItemsSource = painLocation;
            if (migraneVM.getPainLocation().Count != 0 &&
!migraneVM.getPainLocation().Contains(" "))
            {
                selectedPainLocation.AddRange(migraneVM.getPainLocation());
                showListView.ItemsSource = migraneVM.getPainLocation();
            }
        }
        /// <summary>
        /// This method adds to a list of pain locations
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void addToList(object sender, EventArgs args)
        {
            Button button = (Button)sender;
            if (button.CommandParameter != null)
            {
                string value = button.CommandParameter.ToString();
                if (value != null)
                {
                    selectedPainLocation.Add(value);
                }
            }
            else
```

```
            {
                if (addPainLocation.Text != null)
                {
                    selectedPainLocation.Add(addPainLocation.Text);
                    addPainLocation.Text = "";
                }
            }
            showListView.ItemsSource = null;
            showListView.ItemsSource = selectedPainLocation;

        }
        /// <summary>
        /// This method removes pain locations from the list
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void removeFromList(object sender, EventArgs args)
        {
            Button button = (Button)sender;
            string value = button.CommandParameter.ToString();

            selectedPainLocation.RemoveAll(x => x.StartsWith(value));
            showListView.ItemsSource = null;
            showListView.ItemsSource = selectedPainLocation;
        }
        /// <summary>
        /// Returns to previous menu
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private async void returnToMenu(object sender, EventArgs args)
        {
            await Navigation.PopAsync();
        }
        /// <summary>
        /// This method saves the pain locations list to a view model
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private async void saveTypes(object sender, EventArgs args)
        {
            if (selectedPainLocation.Count != 0)
            {
                migraneVM.getPainLocation().Clear();
                migraneVM.setPainLocation(selectedPainLocation);
                await Navigation.PopModalAsync();
            }
            else
            {
                await DisplayAlert("Alert", "You Have Not Selected Any Pain
Locations", "OK");
            }
        }
    }
}
```

4.3.31 RecordSymptoms.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordSymptoms"
BackgroundColor="#005248">


    <StackLayout>
        <StackLayout>
            <ListView x:Name="SymptomsListView" ItemsSource="{Binding symptoms}"
Margin="20" SeparatorVisibility="Default">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Add"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="addToList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>


        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.50*" />
                <ColumnDefinition Width="0.50*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="40" />
            </Grid.RowDefinitions>

            <Entry x:Name="addSymptoms"
                   Grid.Row="0" Grid.Column="0"
               Placeholder="Add Another"
               PlaceholderColor="White"
               TextColor="White"
               VerticalOptions="Center"/>
```

```
            <Button
               Grid.Row="0"
                 Grid.Column="1"
                BackgroundColor="#00905E"
                Text="ADD"
                TextColor="Black"
                CornerRadius="10"
                Clicked="addToList"/>

        </Grid>

        <Label Text="Symptoms" HorizontalTextAlignment="Center" FontSize="Medium"/>

        <StackLayout>
            <ListView  HeightRequest="500" x:Name="showListView" ItemsSource="{Binding
selectedSymptom}" Margin="20">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Remove"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="removeFromList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>



        <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="SAVE"
                Margin="70,5,70,5"
                TextColor="Black"
                Clicked="saveTypes"
                CornerRadius="10" />

    </StackLayout>
</ContentPage>
```

4.3.32 RecordSymptoms.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordSymptoms : ContentPage
    {
        List<string> selectedSymptoms = new List<string>();
        string[] symptoms;
        RecordMigraineViewModel migraneVM;
        internal RecordSymptoms(RecordMigraineViewModel migraneVM)
        {
            InitializeComponent();
            this.migraneVM = migraneVM;
        }
        protected async override void OnAppearing()
        {
            base.OnAppearing();
            symptoms = new[] { "Pounding Pain","Throbbing Pain","Nausea","Light
Sensitivity", "Noise Sensitivity","Vomiting","None" };
            SymptomsListView.ItemsSource = symptoms;
            if (migraneVM.getSymptoms().Count != 0 &&
!migraneVM.getSymptoms().Contains(" "))
            {
                selectedSymptoms.AddRange(migraneVM.getSymptoms());
                showListView.ItemsSource = migraneVM.getSymptoms();
            }
        }
        /// <summary>
        /// This method adds to a list of symptoms
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void addToList(object sender, EventArgs args)
        {
            Button button = (Button)sender;
            if (button.CommandParameter != null)
            {
                string value = button.CommandParameter.ToString();
                if (value != null)
                {
                    selectedSymptoms.Add(value);
                }
            }
            else
```

```
            {
                if (addSymptoms.Text != null)
                {
                    selectedSymptoms.Add(addSymptoms.Text);
                    addSymptoms.Text = "";
                }
            }
            showListView.ItemsSource = null;
            showListView.ItemsSource = selectedSymptoms;

        }
        /// <summary>
        /// This method removes symptoms from the list
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void removeFromList(object sender, EventArgs args)
        {
            Button button = (Button)sender;
            string value = button.CommandParameter.ToString();

            selectedSymptoms.RemoveAll(x => x.StartsWith(value));
            showListView.ItemsSource = null;
            showListView.ItemsSource = selectedSymptoms;
        }
        /// <summary>
        /// Returns to previous menu
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private async void returnToMenu(object sender, EventArgs args)
        {
            await Navigation.PopAsync();
        }
        /// <summary>
        /// This method saves the symptom list to a view model
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private async void saveTypes(object sender, EventArgs args)
        {
            if (selectedSymptoms.Count != 0)
            {
                migraneVM.getSymptoms().Clear();
                migraneVM.setSymptoms(selectedSymptoms);
                await Navigation.PopModalAsync();
            }
            else
            {
                await DisplayAlert("Alert", "You Have Not Selected Any Pain
Locations", "OK");
            }
        }
    }
}
```

4.3.33 RecordTime.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordTime" BackgroundColor="#005248">



    <StackLayout Margin="10">
        <Label Text="Choose start and end time of Migraine Attack"
               Style="{DynamicResource TitleStyle}"
               Margin="0, 20"
               TextColor="White"
               HorizontalTextAlignment="Center"/>

        <Label Text="Start Time:"
               FontAttributes="Bold"
               TextColor="White"
               />

        <TimePicker x:Name="start"

                    Margin="30, 0, 0, 30"
                    TextColor="White"
                    Format="HH:mm:ss" />

        <Label Text="End Time:"
               FontAttributes="Bold"
               TextColor="White"
               />

        <TimePicker x:Name="end"
                    Margin="30, 0, 0, 30"
                    TextColor="White"
                    Format="HH:mm:ss"
                     />

        <StackLayout Orientation="Horizontal"
                     Margin="0, 0, 0, 30">
            <Label Text="Record Length Of Migraine Attack? "
                   VerticalOptions="Center"
                   FontAttributes="Bold"
                   TextColor="White"/>

            <Switch x:Name="includeSwitch"
                    Toggled="OnSwitchToggled"
                    />
        </StackLayout>



        <Label x:Name="resultLabel"
               FontAttributes="Bold"
               TextColor="White"
               HorizontalTextAlignment="Center" />
```

```
        <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Text="SAVE"
                  Grid.Row="5"
                Grid.Column="1"
                TextColor="Black"
                Clicked="recordButton_Clicked"

                VerticalOptions="Center"
                CornerRadius="10" />


    </StackLayout>
</ContentPage>
```

4.3.34 RecordTime.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.ViewModels;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordTime : ContentPage
    {
        RecordMigraneViewModel mvm;
        string lengthOfAttack = "";
        internal RecordTime(RecordMigraneViewModel migraneVM)
        {
            InitializeComponent();
            mvm = migraneVM;
            if (migraneVM.StartTimeOfMigraine != " " && migraneVM.StartTimeOfMigraine
!= null)
            {
                start.Time = TimeSpan.Parse(migraneVM.StartTimeOfMigraine);
            }
            else
            {
                start.Time = DateTime.Now.TimeOfDay;
            }
            if (migraneVM.EndTimeOfMigraine != " " && migraneVM.EndTimeOfMigraine !=
null)
            {

                start.Time = TimeSpan.Parse(migraneVM.EndTimeOfMigraine);
            }
            else
            {
                end.Time = DateTime.Now.TimeOfDay;
            }
        }
        void OnSwitchToggled(object sender, ToggledEventArgs args)
        {
            Recalculate();
        }

        /// <summary>
        /// calculate lenght of time between start and end date
        /// Reference: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-
interface/timepicker
        /// </summary>
        void Recalculate()
        {
            var startTime = start.Time;

            var endTime = end.Time;
            var timeDiff = endTime.Subtract(startTime);
            if (startTime > endTime)
```

```
            {
                timeDiff = timeDiff.Add(TimeSpan.FromHours(12));
            }
            resultLabel.Text = String.Format("Length Of Migraine Attack: {0} Hours {1}
Minutes {2} Seconds", timeDiff.Hours,timeDiff.Minutes,timeDiff.Seconds );
            lengthOfAttack = timeDiff.Hours + ":" + timeDiff.Minutes + ":" +
timeDiff.Seconds;
        }


        private async void returnToMenu(object sender, EventArgs args)
        {
            await Navigation.PopAsync();
        }

        private async void recordButton_Clicked(object sender, EventArgs e)
        {
            Recalculate();
            mvm.LengthOfMigraineAttack = lengthOfAttack;
            mvm.StartTimeOfMigraine = start.Time.ToString();
            mvm.EndTimeOfMigraine = end.Time.ToString();
            await Navigation.PopModalAsync();
        }
    }
}
```

4.3.35 RecordTriggers.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.RecordTriggers"
BackgroundColor="#005248">


    <StackLayout>
        <StackLayout>
            <ListView x:Name="TriggersListView" ItemsSource="{Binding triggers}"
Margin="20" SeparatorVisibility="Default">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Add"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="addToList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>

        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.50*" />
                <ColumnDefinition Width="0.50*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="40" />
            </Grid.RowDefinitions>

            <Entry x:Name="addTriggers"
                   Grid.Row="0" Grid.Column="0"
               Placeholder="Add Another"
               PlaceholderColor="White"
               TextColor="White"
               VerticalOptions="Center"/>
```

```
            <Button
                Grid.Row="0"
                 Grid.Column="1"
                BackgroundColor="#00905E"
                Text="ADD"
                TextColor="Black"
                CornerRadius="10"
                Clicked="addToList"/>

        </Grid>
        <Label Text="Triggers" HorizontalTextAlignment="Center" FontSize="Medium"/>

        <StackLayout>
            <ListView  HeightRequest="500" x:Name="showListView" ItemsSource="{Binding
selectedTriggers}" Margin="20">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="40" />
                                </Grid.RowDefinitions>
                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="White"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>
                                <Button
                                Grid.Row="0"
                                Grid.Column="1"
                                BackgroundColor="#00905E"
                                Text="Remove"
                                TextColor="Black"
                                CornerRadius="10"
                                Clicked="removeFromList"
                                CommandParameter="{Binding .}"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>



        <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="SAVE"
                TextColor="Black"
                Clicked="saveTypes"
                 Margin="70,5,70,5"
                CornerRadius="10" />

    </StackLayout>
</ContentPage>
```

4.3.36 RecordTriggers.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordTriggers : ContentPage
    {
        List<string> selectedTriggers = new List<string>();
        string[] triggers;
        RecordMigraineViewModel migraineVM;
        internal RecordTriggers(RecordMigraineViewModel migraineVM)
        {
            InitializeComponent();
            this.migraineVM = migraineVM;
        }
        protected async override void OnAppearing()
        {
            base.OnAppearing();
            triggers = new[] { "Stress","Lack Of Sleep","Anxiety","Skipped
Meal","Caffeine","Dehydration ","Strong Smell","Neck Pain","Not Sure" };
            TriggersListView.ItemsSource = triggers;
            if (migraineVM.getTriggers().Count != 0 &&
!migraineVM.getTriggers().Contains(" "))
            {
                selectedTriggers.AddRange(migraineVM.getTriggers());
                showListView.ItemsSource = migraineVM.getTriggers();
            }
        }
        /// <summary>
        /// This method adds to a list of triggers
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void addToList(object sender, EventArgs args)
        {
            Button button = (Button)sender;
            if (button.CommandParameter != null)
            {
                string value = button.CommandParameter.ToString();
                if (value != null)
                {
                    selectedTriggers.Add(value);
                }
            }
            else
            {
                if (addTriggers.Text != null)
                {
```

```
                selectedTriggers.Add(addTriggers.Text);
                addTriggers.Text = "";
            }
        }
        showListView.ItemsSource = null;
        showListView.ItemsSource = selectedTriggers;

    }
    /// <summary>
    /// This method removes triggers from the list
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="args"></param>
    private void removeFromList(object sender, EventArgs args)
    {
        Button button = (Button)sender;
        string value = button.CommandParameter.ToString();

        selectedTriggers.RemoveAll(x => x.StartsWith(value));
        showListView.ItemsSource = null;
        showListView.ItemsSource = selectedTriggers;
    }
    /// <summary>
    /// Returns to previous menu
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="args"></param>
    private async void returnToMenu(object sender, EventArgs args)
    {
        await Navigation.PopAsync();
    }
    /// <summary>
    /// This method saves the trigger list to a view model
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="args"></param>
    private async void saveTypes(object sender, EventArgs args)
    {
        if (selectedTriggers.Count != 0)
        {
            migraneVM.getTriggers().Clear();
            migraneVM.setTriggers(selectedTriggers);
            await Navigation.PopModalAsync();
        }
        else
        {
            await DisplayAlert("Alert", "You Have Not Selected Any Triggers",
"OK");
        }
    }
}
}
```

4.3.37 RecordWeather.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-
namespace:MigraineTrackingApp.ViewModels"
             x:Class="MigraineTrackingApp.View.RecordWeather"
BackgroundColor="#005248"
              NavigationPage.HasNavigationBar="false">

    <ContentPage.Resources>
        <ResourceDictionary>
            <Style x:Key="labelStyle"
                   TargetType="Label">
                <Setter Property="FontSize"
                        Value="Small" />
                <Setter Property="TextColor"
                        Value="#0865C0" />
            </Style>

            <Style x:Key="labelResultStyle"
                   TargetType="Label">
                <Setter Property="FontSize"
                        Value="Medium" />
                <Setter Property="Margin"
                        Value="10,0,0,0" />
            </Style>


        </ResourceDictionary>
    </ContentPage.Resources>

    <StackLayout>
        <Grid BackgroundColor="#00905E"
              Padding="10,20,10,10">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>

            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>

            <Label Text="Search by Town/City"
                   Grid.ColumnSpan="3"
                   HorizontalOptions="Center"
                   TextColor="White"
                   FontAttributes="Bold"
                   FontSize="Medium" />

            <Label
                   Text="City:"
                   Grid.Row="1"
                   VerticalOptions="Center"
                   Style="{StaticResource labelStyle}"
                   TextColor="White" />
```

```
            <Entry x:Name="_cityEntry"
                   Grid.Row="1"
                   Grid.Column="1"
                   Margin="5,0"
                   VerticalOptions="Center"
                   BackgroundColor="#68C551"
                   TextColor="Black"
                    />

            <Button Grid.Row="1"
                    Grid.Column="2"
                    Text="Get Weather"

                    BackgroundColor="#00AB58"
                    TextColor="Black"
                    CornerRadius="10"
                    Clicked="OnGetWeatherButtonClicked" />
        </Grid>

        <ScrollView>
            <StackLayout Padding="10">
                <Label Text="Location"
                       TextColor="White"
                       Style="{StaticResource labelStyle}" />
                <Label x:Name="Location" Text="{Binding Title}"
                       TextColor="White"
                       Style="{StaticResource labelResultStyle}" />

                <Label Text="Temperature (°C)"
                       TextColor="White"
                       Style="{StaticResource labelStyle}" />
                <Label x:Name="Temperature" Text="{Binding Main.Temperature}"
                       TextColor="White"
                       Style="{StaticResource labelResultStyle}" />

                <Label Text="Humidity (%)"
                       TextColor="White"
                       Style="{StaticResource labelStyle}" />
                <Label x:Name="Humidity" Text="{Binding Main.Humidity}"
                       TextColor="White"
                       Style="{StaticResource labelResultStyle}" />

            </StackLayout>
        </ScrollView>
        <Button x:Name="saveButton"
                BackgroundColor="#00AB58"
                Grid.Row="5"
                Grid.Column="1"
                Text="RECORD WEATHER"
                TextColor="Black"
                CornerRadius="10"
                Margin="70,20,70,20"
                Clicked="recordButton_Clicked"/>
    </StackLayout>
</ContentPage>
```

4.3.38 RecordWeather.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Models;
using MigraineTrackingApp.ViewModels;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RecordWeather : ContentPage
    {
        RestService _restService;
        string OpenWeatherMapEndpoint =
"https://api.openweathermap.org/data/2.5/weather";
        string OpenWeatherMapAPIKey = "1cdddc65df6697af0fd14a9ccae9b7c1";
        RecordMigraneViewModel mVm;
        internal RecordWeather(RecordMigraneViewModel migraneVM)
        {
            InitializeComponent();
            mVm = migraneVM;
            if(migraneVM.Location != " " && migraneVM.Location != null)
            {
                Location.Text = migraneVM.Location;
            }
            if (migraneVM.Humidity != " " && migraneVM.Humidity != null)
            {
                Humidity.Text = migraneVM.Humidity;
            }
            if (migraneVM.Temperature != " " && migraneVM.Temperature != null)
            {
                Temperature.Text = migraneVM.Temperature;
            }
            _restService = new RestService();
        }
        async void OnGetWeatherButtonClicked(object sender, EventArgs e)

        // Reference : https://github.com/xamarin/xamarin-forms-
samples/tree/main/Weather
        {
            if (!string.IsNullOrWhiteSpace(_cityEntry.Text))
            {
                WeatherData weatherData = await
_restService.GetWeatherData(GenerateRequestUri(OpenWeatherMapEndpoint));
                BindingContext = weatherData;
            }

            else
            {
                await DisplayAlert("", "Please Enter A Valid Location", "OK");
            }
        }
```

```
        string GenerateRequestUri(string endpoint)
        {
            string requestUri = endpoint;
            requestUri += $"?q={_cityEntry.Text}";
            requestUri += "&units=metric"; // or units=metric
            requestUri += $"&APPID={OpenWeatherMapAPIKey}";
            return requestUri;
        }

        private async void recordButton_Clicked(object sender, EventArgs e)
        {
            mVm.Location = Location.Text;
            mVm.Humidity = Humidity.Text;
            mVm.Temperature = Temperature.Text;

            await Navigation.PopModalAsync();
        }
    }
}
```

4.3.39 SelectMonth.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.SelectMonth"
                   xmlns:magic="clr-namespace:MagicGradients;assembly=MagicGradients">


    <Grid>

        <magic:GradientView VerticalOptions="FillAndExpand">
            <magic:GradientView.GradientSource>
                <magic:LinearGradient Angle="180">
                    <magic:GradientStop Color="#00AB58" Offset="0" />
                    <magic:GradientStop Color="#F3F354" Offset="1" />
                </magic:LinearGradient>
            </magic:GradientView.GradientSource>
        </magic:GradientView>


        <StackLayout>
            <Label Text="Select A Month"   FontAttributes="Bold"
HorizontalTextAlignment="Center"  TextColor="Black"

       TextTransform="Uppercase" />
            <ListView x:Name="listView" ItemsSource="{Binding records}"
HasUnevenRows="True">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout Orientation="Vertical">
                                <Grid Margin="5,20,30,10">


                                    <Grid.ColumnDefinitions>
                                        <ColumnDefinition Width="0.50*" />
                                        <ColumnDefinition Width="0.50*" />
                                    </Grid.ColumnDefinitions>


                                    <Grid.RowDefinitions>
                                        <RowDefinition Height="49" />
                                    </Grid.RowDefinitions>


                                    <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="Black"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>


                                    <Button Grid.Row="0" Grid.Column="1"
                                    Text="Select"
                                    CornerRadius="10" Margin="5, 0,0,5"
                                    FontAttributes="Bold"
```

```
                                        BackgroundColor="#014941"

                    TextColor="white"
                                    Clicked="displayRecord"
                                    CommandParameter="{Binding .}" />
                                </Grid>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </Grid>
</ContentPage>
```

4.3.40 SelectMonth.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Models;
using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SelectMonth : ContentPage
    {
        List<string> months;
        string seletedMonth;
        List<Migraine> allRecords;
        List<string> medicationForTheMonth =  new List<string>();
        private ShowMigraineRecordsViewModel vm = new ShowMigraineRecordsViewModel();
        public SelectMonth(List<Migraine> allRecords,List<string> months)
        {
            InitializeComponent();
            this.months = months;
            this.allRecords = allRecords;
        }
        protected async override void OnAppearing()
        {
            listView.ItemsSource = months;
            base.OnAppearing();
        }
        /// <summary>
        /// when month selected pain intensity info and date/time are stored in a list
of display graphs and mediction info stored in a string list
        /// display chart for pain intensity and medication taken for the month
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private async void displayRecord(object sender, EventArgs e)
        {
            Button button = (Button)sender;
            seletedMonth = button.CommandParameter.ToString();
            List<DisplayGraph>recordsForThatMonth =
vm.getRecordsForThatMonth(allRecords,seletedMonth,ref medicationForTheMonth);

            await Navigation.PushModalAsync(new
DisplayPainIntensityChart(recordsForThatMonth, medicationForTheMonth));
        }
    }
}
```

4.3.41 ShowMigraineDetails.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.ShowMigraineDetails"
             xmlns:magic="clr-namespace:MagicGradients;assembly=MagicGradients">


    <Grid>

        <magic:GradientView VerticalOptions="FillAndExpand">
            <magic:GradientView.GradientSource>
                <magic:LinearGradient Angle="180">
                    <magic:GradientStop Color="#00AB58" Offset="0" />
                    <magic:GradientStop Color="#F3F354" Offset="1" />
                </magic:LinearGradient>
            </magic:GradientView.GradientSource>
        </magic:GradientView>



       <ScrollView>
        <StackLayout>

                <Label x:Name="title" Text="Migraine Record" FontSize="Title"
FontAttributes="Bold" HorizontalTextAlignment="Center"  TextColor="Black" />

                <Label x:Name="startDateHeading" Text="Start Date" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>

                <Label x:Name="startD" Text="" TextColor="Black"/>

                <Label x:Name="endDateHeading" Text="End Date" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>

                <Label x:Name="endD" Text="" TextColor="Black"/>

                <Label x:Name="startTimeHeading" Text="Start Time" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>

                <Label x:Name="startT" Text="" TextColor="Black"/>

                <Label x:Name="endTimeHeading" Text="End Time" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>


                <Label x:Name="endT" Text="" TextColor="Black"/>

                <Label x:Name="humidityHeading" Text="Humidity (%)"
FontSize="Subtitle" FontAttributes="Bold" TextDecorations="Underline"
TextColor="Black" />


                <Label x:Name="humd" Text="" TextColor="Black"/>

                <Label x:Name="tempHeading" Text="Temperature (°C)"
FontSize="Subtitle" FontAttributes="Bold" TextDecorations="Underline"
TextColor="Black"/>
```

```
                <Label x:Name="temp" Text="" TextColor="Black"/>


                <Label x:Name="locationHeading" Text="Location" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>

                <Label x:Name="loc" Text="" TextColor="Black"/>

                <Label x:Name="durationHeading" Text="Migraine Duration (H:M:S)"
FontSize="Subtitle" FontAttributes="Bold" TextDecorations="Underline"
TextColor="Black"/>


                <Label x:Name="migraineD" Text="" TextColor="Black"/>

                <Label x:Name="painIntensityHeading" Text="Pain Intensity"
FontSize="Subtitle" FontAttributes="Bold" TextDecorations="Underline"
TextColor="Black"/>

                <Label x:Name="painI" Text="" TextColor="Black"/>

                <Label x:Name="fdHeading" Text="Food Eaten" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>


                <Label x:Name="food" Text="" TextColor="Black"/>

                <Label x:Name="typeHeading" Text="Migraine Type" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>

                <Label x:Name="migraineT" Text="" TextColor="Black"/>

                <Label x:Name="medHeading" Text="Medication" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>

                <Label x:Name="medicationT" Text="" TextColor="Black"/>


                <Label x:Name="painLocHeading" Text="Pain Location"
FontSize="Subtitle" FontAttributes="Bold" TextDecorations="Underline"
TextColor="Black" />


                <Label x:Name="painLoc" Text="" TextColor="Black"/>

                <Label x:Name="sympHeading" Text="Symptoms" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>


                <Label x:Name="symptom" Text="" TextColor="Black"/>

                <Label x:Name="trigHeading" Text="Triggers" FontSize="Subtitle"
FontAttributes="Bold" TextDecorations="Underline" TextColor="Black"/>


                <Label x:Name="trigger" Text="" TextColor="Black"/>

            <Button Text="Email Record"
                    Margin="70,20,70,20"
                    BackgroundColor="#014941"
                    VerticalOptions="Center"
                     TextColor="white"
                    CornerRadius="10"
                Clicked="OnButtonClicked" />
```

```
            <Button Text="Update Record"
                VerticalOptions="Center"
                   Margin="70,20,70,20"
                    BackgroundColor="#014941"
                     TextColor="white"
                    CornerRadius="10"
                Clicked="updateClicked" />


            </StackLayout>
        </ScrollView>
    </Grid>
</ContentPage>
```

4.3.42 ShowMigraineDetails.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Models;
using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ShowMigraineDetails : ContentPage
    {
        Migraine migraine;
        RecordMigraneViewModel vm = new RecordMigraneViewModel();
        string foodList = "";
        string migraineTypeList = "";
        string medicationTypeList = "";
        string painLocationList = "";
        string symptomsList = "";
        string triggersList = "";
        string email = "";
        string emailMesage = "";
        string userId = "";
        string fList = "";
        string migList = "";
        string medList = "";
        string pList = "";
        string sList = "";
        string tList = "";

        IAuth auth;
       /// <summary>
       /// setting migraine record values
       /// </summary>
       /// <param name="record"></param>
       /// <param name="email"></param>
       /// <param name="id"></param>
       /// <param name="auth"></param>
        public ShowMigraineDetails(Migraine record,string email,string id, IAuth auth)
        {
            migraine = record;
            vm.StartDate = migraine.startDate;
            vm.EndDate = migraine.endDate;
            vm.StartTimeOfMigraine = migraine.startTime;
            vm.EndTimeOfMigraine = migraine.endTime;
            vm.Humidity = migraine.humidity;
            vm.Location = migraine.location;
            vm.PainIntensity = migraine.painIntensity;
            vm.LengthOfMigraineAttack = migraine.migraineDuration;
            vm.Temperature = migraine.temperature;
            // if null value found in a migraine record detail ignore/ dont fill in
```

```
            if(migraine.migraineType != null)
            {
                vm.setMigraneTypes(migraine.migraineType);
            }
            if (migraine.triggers != null)
            {
                vm.setTriggers(migraine.triggers);
            }
            if (migraine.symptoms != null)
            {
                vm.setSymptoms(migraine.symptoms);
            }
            if (migraine.foods != null)
            {
                vm.setFoodEaten(migraine.foods);
            }
            if (migraine.painLocation != null)
            {
                vm.setPainLocation(migraine.painLocation);
            }
            if (migraine.medicationType != null)
            {
                vm.setMedicationTypes(migraine.medicationType);
            }
            InitializeComponent();
            this.email = email;
            userId = id;
            this.auth = auth;
        }
        /// <summary>
        /// content of email
        /// </summary>
        protected async override void OnAppearing()
        {

            startD.Text = migraine.startDate;
            emailMesage += "Start Date: "+ startD.Text + "\n";

            endD.Text = migraine.endDate;
            emailMesage += "\n" +"End Date: "+endD.Text + "\n";


            startT.Text = migraine.startTime;
            emailMesage += "\n" + "Start Time: " + startT.Text + "\n";


            endT.Text = migraine.endTime;
            emailMesage += "\n" + "End Time: " + endT.Text + "\n";


            humd.Text = migraine.humidity;
            emailMesage += "\n" + "Humidity: " +humd.Text + "%" + "\n";

            temp.Text = migraine.temperature;
            emailMesage += "\n" + "Temperature: " +temp.Text + "°C" + "\n";


            loc.Text = migraine.location;
            emailMesage += "\n" + "Location: " + loc.Text + "\n";


            migraineD.Text = migraine.migraineDuration;
            emailMesage += "\n" + "Migraine Duration (H:M:S): " + migraineD.Text +
"\n";
```

```
    painI.Text = migraine.painIntensity;
    emailMesage += "\n" + "Pain Intensity: " + painI.Text + "\n";

    if(migraine.foods != null)
    {
        fList = string.Join("\n ", migraine.foods);
        emailMesage += "\n" + "Food Eaten: " + "\n" + fList + "\n";
    }

    if (migraine.migraineType != null)
    {
        migList = string.Join("\n ", migraine.migraineType);
        emailMesage += "\n" + "Migraine Type: " + migList + "\n";
    }

    if (migraine.medicationType != null)
    {
        medList = string.Join("\n ", migraine.medicationType);
        emailMesage += "\n" + "Medication: " + medList + "\n";
    }
    if (migraine.painLocation != null)
    {
        pList = string.Join("\n ", migraine.painLocation);
        emailMesage += "\n" + "Pain Location: " + pList + "\n";
    }
    if (migraine.symptoms != null)
    {
        sList = string.Join("\n ", migraine.symptoms);
        emailMesage += "\n" + "Symptoms: " + sList + "\n";
    }
    if (migraine.triggers != null)
    {
        tList = string.Join("\n ", migraine.triggers);
        emailMesage += "\n" + "Triggers: " + tList + "\n";
    }

    food.Text = fList;
    medicationT.Text = medList;
    migraineT.Text = migList;
    painLoc.Text = pList;
    symptom.Text = sList;
    trigger.Text = tList;
    base.OnAppearing();
}
/// <summary>
/// sends email with migraine record details
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private async void OnButtonClicked(object sender, EventArgs e)
{
    List<string> recipients = new List<string>();
    recipients.Add(email);
    await sendMigraineRecord(recipients, emailMesage);
}
/// <summary>
/// goes to record migraine page to allow user to update record
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private async void updateClicked(object sender, EventArgs e)
{
```

```
            await Navigation.PushModalAsync(new RecordMigraine(userId,vm,email,
auth));
        }
        /// <summary>
        /// this method send the email using Xamarin Essentails email API
        /// </summary>
        /// <param name="recipients"></param>
        /// <param name="body"></param>
        /// <returns></returns>
         public async Task<bool> sendMigraineRecord(List<string> recipients,string
body)
        {
            try
            {
                EmailMessage message = new EmailMessage
                {
                    Subject = "Migraine Record",
                    Body = body,
                    To = recipients,
                };
                await Email.ComposeAsync(message);
                return true;
            }
            catch (FeatureNotSupportedException ns) //if phone does not support this
API
            {
                var value = ns.StackTrace;
                Console.WriteLine(value);
                return false;
            }
            catch (Exception ex)
            {
                var value = ex.StackTrace;
                Console.WriteLine(value);
                return false;
            }
        }
    }
}
```

4.3.43 ShowPreviousRecords.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.showPreviousRecords"
             xmlns:magic="clr-namespace:MagicGradients;assembly=MagicGradients">


    <Grid>

        <magic:GradientView VerticalOptions="FillAndExpand">
            <magic:GradientView.GradientSource>
                <magic:LinearGradient Angle="180">
                    <magic:GradientStop Color="#007361" Offset="0" />
                    <magic:GradientStop Color="#F3F354" Offset="1" />
                </magic:LinearGradient>
            </magic:GradientView.GradientSource>
        </magic:GradientView>

        <StackLayout>
            <Label Text="Select A Previous Migraine Record"
HorizontalTextAlignment="Center" TextColor="#005248" FontAttributes="Bold"

       TextTransform="Uppercase" />
        <ListView x:Name="listView" ItemsSource="{Binding records}"
HasUnevenRows="True">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout Orientation="Vertical">
                            <Grid Margin="5,20,30,10">



                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="0.50*" />
                                    <ColumnDefinition Width="0.50*" />
                                </Grid.ColumnDefinitions>



                                <Grid.RowDefinitions>
                                    <RowDefinition Height="49" />
                                </Grid.RowDefinitions>



                                <Label Grid.Row="0" Grid.Column="0"
                                   TextColor="Black"
                                   FontSize="Medium" FontFamily="Arial"
                                   HorizontalTextAlignment="Center"
                                   Text="{Binding .}"/>



                                    <Button Grid.Row="0" Grid.Column="1"
                                            TextColor="white"
                                    Text="Select"
                                    CornerRadius="10" Margin="5, 0,0,5"
```

```
                                    FontAttributes="Bold"
                                        BackgroundColor="#014941"
                                    Clicked="displayRecord"
                                    CommandParameter="{Binding .}" />
                            </Grid>
                        </StackLayout>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
        </StackLayout>

    </Grid>

</ContentPage>
```

4.3.44 ShowPreviousRecords.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Models;
using MigraineTrackingApp.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class showPreviousRecords : ContentPage
    {
        List<Migraine> allRecords;
        string selectedDate;
        private string userId;
        Migraine recordAtADate;
        string email = "";
        IAuth auth;
        /// <summary>
        ///
        /// </summary>
        /// <param name="id"></param>
        /// <param name="email"></param>
        /// <param name="mig"></param>
        /// <param name="auth"></param>
        public showPreviousRecords(string id,string email, List<Migraine> mig, IAuth
auth)
        {
            InitializeComponent();
            userId = id;
            this.email = email;
            allRecords = mig;
            this.auth = auth;
        }
        protected async override void OnAppearing()
        {
            var records = allRecords.Select(i => i.dateEntered).ToList();
            listView.ItemsSource = records;
            base.OnAppearing();
        }
       /// <summary>
       /// gets selected date from all records passes to show migraine detail page
       ///
       /// </summary>
       /// <param name="sender"></param>
       /// <param name="e"></param>
        private async void displayRecord(object sender, EventArgs e)
        {
```

```
            Button button = (Button)sender;
            selectedDate = button.CommandParameter.ToString();
            foreach(Migraine record in allRecords)
            {
                if(record.dateEntered == selectedDate)
                {
                    recordAtADate = record;
                    break;
                }
            }

            await Navigation.PushModalAsync(new ShowMigraineDetails(recordAtADate,
email, userId, auth));
        }
    }
}
```

4.3.45 UpdateProfile.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MigraineTrackingApp.View.UpdateProfile"
              xmlns:magic="clr-namespace:MagicGradients;assembly=MagicGradients">
    <Grid>

        <magic:GradientView VerticalOptions="FillAndExpand">
            <magic:GradientView.GradientSource>
                <magic:LinearGradient Angle="180">
                    <magic:GradientStop Color="#007361" Offset="0" />
                    <magic:GradientStop Color="#F3F354" Offset="1" />
                </magic:LinearGradient>
            </magic:GradientView.GradientSource>
        </magic:GradientView>

        <ScrollView VerticalScrollBarVisibility="Always">

            <StackLayout Padding="0,50,0,0">

                <Image x:Name="logoImage"
               Aspect="AspectFit"
               HorizontalOptions="FillAndExpand"
               HeightRequest="250"
               Margin="20,0,10,50"/>

                <Image x:Name="resultImage" HeightRequest="50" WidthRequest="50"/>



                <Label Text="First Name" FontSize="Default" TextColor="white"
FontAttributes="Bold"/>

                <Entry x:Name="memberFirstName"
               Placeholder="First Name"
               PlaceholderColor="#014941"
               TextColor="#014941"
               VerticalOptions="Center"
               />


                <Label Text= "Gender" FontSize="Default" TextColor="white"
FontAttributes="Bold"/>


                <StackLayout Margin="0,0,0,0">
                    <StackLayout Orientation="Horizontal">

                        <RadioButton Content="Male"
                          CheckedChanged="getGender"
                                 TextColor="#014941"/>
                        <RadioButton Content="Female"
                          CheckedChanged="getGender"
                         TextColor="#014941"/>
                        <RadioButton Content="Non Binary"
                          CheckedChanged="getGender"
                         TextColor="#014941"/>
                    </StackLayout>
                </StackLayout>
```

```
                <Label Text="DOB" FontSize="Default" TextColor="white"
FontAttributes="Bold"/>

                <DatePicker x:Name="memberDob"
                        Format="D"
                            TextColor="#014941"/>



                <Button Text="Save Profile"
                    Clicked="updateInfo"
                    BackgroundColor="#014941"
                     VerticalOptions="Center"
                     CornerRadius="10"

                    TextColor="white"
                    Margin="70,20,70,20"
                    />

            </StackLayout>
        </ScrollView>
    </Grid>
</ContentPage>
```

4.3.46 UpdateProfile.xaml.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.ViewModels;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace MigraineTrackingApp.View
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class UpdateProfile : ContentPage
    {
        MemberViewModel memberVm = new MemberViewModel();
        string Id = "";
        private RadioButton button;
        private string gender = "";
        string firstName = "";
        string dob = "";

        public UpdateProfile(string memberFirstName,string memberDob,string gen,string
userId)
        {
            InitializeComponent();
            this.Id = userId;
            firstName = memberFirstName;
            dob = memberDob;
            this.gender = gen;
            var assemble = typeof(ProfilePage);

            logoImage.Source =
ImageSource.FromResource("MigraineTrackingApp.Assets.Images.logo.png", assemble);
        }
        protected async override void OnAppearing()
        {
            base.OnAppearing();
            memberFirstName.Text = firstName;
            if(dob != " ")
            {
                memberDob.Date = DateTime.Parse(dob);
            }
        }
        /// <summary>
        /// This button gets the gender from the selected radio button
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void getGender(object sender, CheckedChangedEventArgs e)
        {
            button = sender as RadioButton;
            gender = button.Content.ToString();
        }

        /// <summary>
        /// This button sends the firstname DOB and gender to database
        /// </summary>
```

```
        /// <param name="sender"></param>
        /// <param name="e"></param>
        async void updateInfo(System.Object sender, System.EventArgs e)
        {
            string dateAndTime = memberDob.Date.ToString();
            int spacePosition = dateAndTime.IndexOf(" ");
            string dateOnly = dateAndTime.Substring(0,spacePosition);
            memberVm.createProfile(memberFirstName.Text, dateOnly, gender, Id);
            await Navigation.PopModalAsync();
        }
    }
}
```

4.4 ViewModels

4.4.1 IAuth.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

namespace MigraineTrackingApp
{ //Reference: https://www.lindseybroos.be/2020/03/xamarin-forms-and-firebase-
authentication/
    /// <summary>
    /// these methods are used to access firebase login, logout and signup methods for
each device - android or iOS
    /// </summary>
    public interface IAuth
    {
        Task<string> LoginWithEmailPassword(string email, string password);

        Task<string> SignupWithEmailPassword(string email, string password);
        void Logout();
    }
}
```

4.4.2 MemberViewModel.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using MigraineTrackingApp.Models;
using MigraineTrackingApp.Services;
using System.Threading.Tasks;

namespace MigraineTrackingApp.ViewModels
{
    class MemberViewModel
    {
        DbConnect db;

        public MemberViewModel()
        {
            db = new DbConnect(); //connection to DB
        }
        /// <summary>
        /// Get member
        /// </summary>
        /// <param name="userId"></param>
        /// <returns></returns>
        public async Task<Member> getMember(string userId)
        {
            var result = await db.GetMember(userId);
            return result;

        }
        /// <summary>
        /// Send data to create a member
        /// </summary>
        /// <param name="firstName"></param>
        /// <param name="lastName"></param>
        /// <param name="dob"></param>
        /// <param name="gender"></param>
        /// <param name="userid"></param>
        public async void createProfile(string firstName,string dob,string
gender,string userid)
        {
            await db.createProfile(firstName, dob, gender, userid);
        }
    }
}
```

4.4.3 RecordMigraineViewModel.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Services;
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

namespace MigraineTrackingApp.ViewModels
{
    /// <summary>
    /// This class is used to save data belonging to the record migrane usecase
    /// </summary>
    class RecordMigraneViewModel
    {
        List<string> migraneTypes = new List<string>();
        List<string> painLocation = new List<string>();
        List<string> medicationType = new List<string>();
        List<string> symptoms = new List<string>();
        List<string> triggers = new List<string>();
        List<string> foods = new List<string>();
        List<string> allergens = new List<string>();

        List<string> result = new List<string>();

        string location;
        string humidity;
        string temperature;
        string startTime;
        string endTime;
        string startdate;
        string enddate;
        string migraineDuration;
        string painIntensity;

        DbConnect db;
        /// <summary>
        /// make connection to database
        /// </summary>
        public RecordMigraneViewModel()
        {
            db = new DbConnect();
        }
        /// <summary>
        /// This menthod saves the migrane types list
        /// </summary>
        /// <param name="types"></param>
        public void setMigraneTypes(List<string> types)
        {
            migraneTypes.AddRange(types);
        }
        /// <summary>
        /// This method returns the migrane types list
        /// </summary>
        /// <returns></returns>
```

```
        public List<string> getMigraneTypes()
        {
            return migraneTypes;
        }

        ///
////////////////////////////////////////////////////////////////////////////////
/////////



        /// <summary>
        /// This menthod saves the pain locations list
        /// </summary>
        /// <param name="types"></param>
        public void setPainLocation(List<string> types)
        {
            painLocation.AddRange(types);
        }
        /// <summary>
        /// This method returns the pain location  list
        /// </summary>
        /// <returns></returns>
        public List<string> getPainLocation()
        {
            return painLocation;
        }

////////////////////////////////////////////////////////////////////////////////
//////
        ///
        /// <summary>
        /// This menthod saves the pain locations list
        /// </summary>
        /// <param name="types"></param>
        public void setMedicationTypes(List<string> types)
        {
            medicationType.AddRange(types);
        }
        /// <summary>
        /// This method returns the pain location  list
        /// </summary>
        /// <returns></returns>
        public List<string> getMedicationTypes()
        {
            return medicationType;
        }

////////////////////////////////////////////////////////////////////////////////
        ///
        /// <summary>
        /// This menthod saves the pain locations list
        /// </summary>
        /// <param name="types"></param>
        public void setSymptoms(List<string> types)
        {
            symptoms.AddRange(types);
        }
        /// <summary>
        /// This method returns the pain location  list
        /// </summary>
        /// <returns></returns>
        public List<string> getSymptoms()
        {
```

```
            return symptoms;
        }

        ////////////////////////////////////////////////////////////////////
        ///
        /// <summary>
        /// This menthod saves the triggers list
        /// </summary>
        /// <param name="types"></param>
        public void setTriggers(List<string> types)
        {
            triggers.AddRange(types);
        }
        /// <summary>
        /// This method returns the pain location  list
        /// </summary>
        /// <returns></returns>
        public List<string> getTriggers()
        {
            return triggers;
        }

        ////////////////////////////////////////////////////////////////////////

        public string Location
        {
            get => location;
            set => location = value;
        }

        public string Humidity
        {
            get => humidity;
            set => humidity = value;
        }

        public string Temperature
        {
            get => temperature;
            set => temperature = value;
        }


////////////////////////////////////////////////////////////////////////////

        public string StartTimeOfMigraine
        {
            get => startTime;
            set => startTime = value;
        }
        public string EndTimeOfMigraine
        {
            get => endTime;
            set => endTime = value;
        }
        public string LengthOfMigraineAttack
        {
            get => migraineDuration;
            set => migraineDuration = value;
        }

        ////////////////////////////////////////////////////////////////////
        public string StartDate
        {
```

```
            get => startdate;
            set => startdate = value;
        }
        public string EndDate
        {
            get => enddate;
            set => enddate = value;
        }


        public string PainIntensity
        {
            get => painIntensity;
            set => painIntensity = value;
        }

////////////////////////////////////////////////////////////////////////////////

        public void setFoodEaten(List<string> types)
        {
            foods.AddRange(types);
        }

        public List<string> getFoodEaten()
        {
            return foods;
        }

        public void resetFoodList()
        {
            foods.Clear();
        }
        /// <summary>
        /// This method sets all allergens for the food products scanned
        /// </summary>
        /// <param name="types"></param>
        public void setAllAllergenypes(string []allergen)
        {
            for(int i = 0;i < allergen.Length; i++)
            {
                if (allergens.Contains(allergen[i]))
                {

                }
                else
                {
                    allergens.Add(allergen[i]);
                }
            }
        }
        /// <summary>
        /// This method returns all allergens for the food products entered
        /// </summary>
        /// <returns></returns>
        public List<string> getAllergenTypes()
        {
            return allergens;
        }
        /// <summary>
        /// This method adds the current scanned allergen info into the user allergen
list in the database
        /// </summary>
        /// <param name="uid"></param>
        public async void sendAllergenInfo(string uid)
```

```
        {
            bool value = await db.addAllergens(uid, getAllergenTypes());
        }
        /// <summary>
        /// This method returns a list of allergens from the databse for that user
        /// </summary>
        /// <param name="uid"></param>
        public async Task<List<string>> getAllergenList(string uid)
        {
           result = await db.getAllergenListFromDb(uid);
           return result;
        }
        /// <summary>
        /// This method checks if allergens for a product are already in the database
if they are remove them
        /// </summary>
        /// <param name="uid"></param>
        public async void checkIfAllergensAreInDB(string uid)
        {
            result = await getAllergenList(uid);

            if (result != null)
            {
                foreach (var allergen in allergens)
                {
                    if (result.Contains(allergen))
                    {
                        allergens.Remove(allergen);
                    }
                }
            }
        }


       /// <summary>
       /// this method takes user ID and an array of scanned allergens
       /// </summary>
       /// <param name="uid"></param>
       /// <param name="scannedAllergens"></param>
       /// <returns></returns>
        public async Task<string> checkScannedItem(string uid,string[]
scannedAllergens)
        {
            string allergensFound = " ";
            result = await getAllergenList(uid); //gets back allergens from database
            foreach (var allergen in scannedAllergens)
            {
                if (result.Contains(allergen)) // if allergens are already in database
might cause migraine
                {
                    allergensFound += allergen + " ";
                }
            }
            if(allergensFound == " ")
            {
                return "This Food Item Has Not Caused A Previous Migraine";

            }
            return "This Food Item May Cause A Migraine\n Allergens Found: " +
allergensFound + "\n";
        }

       /// <summary>
       /// this sends the migraine details to database
```

```
        /// </summary>
        /// <param name="currentDate"></param>
        /// <param name="uid"></param>
         public async void sendRecordDetailsToDataase(string currentDate,string uid)
         {
             setValuesIfNotFilledIn();
             bool value = await db.createMigraineRecord(uid, getMigraneTypes(),
getPainLocation(), getMedicationTypes(), getSymptoms(), getTriggers(), getFoodEaten(),
Location, Humidity, Temperature, StartTimeOfMigraine, EndTimeOfMigraine, StartDate,
EndDate, LengthOfMigraineAttack, PainIntensity, currentDate);
         }

         /// <summary>
         /// if values have not been filled in sets initial values to empty variable
         /// </summary>

         public void setValuesIfNotFilledIn()
         {
             List<string> ifListEmpty = new List<string>();
             ifListEmpty.Add(" ");
             if (getMigraneTypes().Count == 0)
             {
                 setMigraneTypes(ifListEmpty);
             }
             if (getPainLocation().Count == 0)
             {
                 setPainLocation(ifListEmpty);
             }
             if (getMedicationTypes().Count == 0)
             {
                 setMedicationTypes(ifListEmpty);
             }
             if (getSymptoms().Count == 0)
             {
                 setSymptoms(ifListEmpty);
             }
             if (getTriggers().Count == 0)
             {
                 setTriggers(ifListEmpty);
             }
             if (getFoodEaten().Count == 0)
             {
                 setFoodEaten(ifListEmpty);
             }
             if (Location == null)
             {
                 Location = " ";
             }
             if (Humidity == null)
             {
                 Humidity = " ";
             }
             if (Temperature == null)
             {
                 Temperature = " ";
             }
             if (StartTimeOfMigraine == null)
             {
                 StartTimeOfMigraine = " ";
             }
             if (EndTimeOfMigraine == null)
             {
                 EndTimeOfMigraine = " ";
             }
```

```
            if (StartDate == null)
            {
                StartDate = " ";
            }
            if (EndDate == null)
            {
                EndDate = " ";
            }
            if (PainIntensity == null)
            {
                PainIntensity = " ";
            }
            if (LengthOfMigraineAttack == null)
            {
                LengthOfMigraineAttack = " ";
            }
        }
    }
}
```

4.4.4 RestService.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using MigraineTrackingApp.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

namespace MigraineTrackingApp.ViewModels
{
    /// <summary>
    /// accesses openweather API through http request
    /// </summary>
    public class RestService
    {
        HttpClient _client;

       /// <summary>
       /// create instance of http client
       /// </summary>
        public RestService()
        {
            _client = new HttpClient();
        }

        /// <summary>
        /// query gets back response deserialise JSON response and stores in models
        /// </summary>
        /// <param name="query">contains open weather API and location</param>
        /// <returns></returns>
        public async Task<WeatherData> GetWeatherData(string query)
        {
            WeatherData weatherData = null;
            try
            {
                var response = await _client.GetAsync(query);
                if (response.IsSuccessStatusCode)
                {
                    var content = await response.Content.ReadAsStringAsync();
                    weatherData = JsonConvert.DeserializeObject<WeatherData>(content);
                }
            }
            catch (Exception ex)
            {
                //Debug.WriteLine("\t\tERROR {0}", ex.Message);
            }

            return weatherData;
        }
    }
}
```

4.4.5 ShowMigraineRecordsViewModel.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 * Date: 19/4/2022
 */

using MigraineTrackingApp.Models;
using MigraineTrackingApp.Services;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MigraineTrackingApp.ViewModels
{
    class ShowMigraineRecordsViewModel
    {
        DbConnect db;
        string email;
        public ShowMigraineRecordsViewModel()
        {
            db = new DbConnect();
        }
        /// <summary>
        /// This method returns a list of previous migraine attacks for that user
        /// </summary>
        /// <param name="uid"></param>
        /// <returns></returns>
        public async Task<List<Migraine>> getAllPrevousMigraineRecords(string uid)
        {
            List <Migraine> value = await db.getMigraineRecords(uid);
            return value;
        }
        public string Email
        {
            get => email;
            set => email = value;
        }
       /// <summary>
       /// this method loops through a list of migraines using numeric month value to
build a list of months with months name
       /// </summary>
       /// <param name="objList">list of migraine</param>
       /// <param name="months">list of strings</param>
       /// <returns></returns>
        public List<string> getListOfMonths(List<Migraine> objList,List<string>
months)
        {
            foreach (Migraine obj in objList)
            {
                string monthNumber = obj.dateEntered.Substring(3, 2); // gets month in
numeric value
                switch (monthNumber)
                {
                    case "01":
                        months.Add("January");
                        break;
```

```
                    case "02":
                        months.Add("February");
                        break;
                    case "03":
                        months.Add("March");
                        break;
                    case "04":
                        months.Add("April");
                        break;
                    case "05":
                        months.Add("May");
                        break;
                    case "06":
                        months.Add("June");
                        break;
                    case "07":
                        months.Add("July");
                        break;
                    case "08":
                        months.Add("August");
                        break;
                    case "09":
                        months.Add("September");
                        break;
                    case "10":
                        months.Add("October");
                        break;
                    case "11":
                        months.Add("November");
                        break;
                    case "12":
                        months.Add("December");
                        break;
                }
            }
            List<string> monthList = months.Distinct().ToList();//return no duplicates
            monthList = monthList.OrderBy(s => DateTime.ParseExact(s, "MMMM", new
CultureInfo("en-US"))).ToList();//reference:
https://stackoverflow.com/questions/8539088/sorting-months-in-a-list
            return monthList;
        }
        /// <summary>
        /// gets the records for the chosen months
        /// </summary>
        /// <param name="objList">list of migraine</param>
        /// <param name="selectedMonth">string</param>
        /// <param name="meds">list of strings</param>
        /// <returns></returns>
        public List<DisplayGraph> getRecordsForThatMonth(List<Migraine> objList,string
selectedMonth,ref List<string> meds)
        {
            Dictionary<string, string> months = new Dictionary<string, string>();
            List<DisplayGraph> recordsForThatMonth = new List<DisplayGraph>();
            DisplayGraph graphObj;
            months.Add("January", "01");
            months.Add("February", "02");
            months.Add("March", "03");
            months.Add("April", "04");
            months.Add("May", "05");
            months.Add("June", "06");
            months.Add("July", "07");
            months.Add("August", "08");
            months.Add("September", "09");
            months.Add("October", "10");
```

```
            months.Add("November", "11");
            months.Add("December", "12");
            string monthSelected = months[selectedMonth];
            foreach (Migraine obj in objList)
            {
                if(monthSelected.Equals(obj.dateEntered.Substring(3, 2)) &&
obj.painIntensity != " ")//only look at the selected month entries
                {
                    if(obj.medicationType.Count != 0)//check if the medication list is
empty
                    {
                        foreach(string medication in obj.medicationType)//loop through
all the mdications
                        {
                            if (!meds.Contains(medication) && medication != " ")//add
medications that are not in the list
                            {
                                meds.Add(medication);
                            }
                        }
                    }

                    graphObj = new DisplayGraph();
                    graphObj.Date = obj.dateEntered; // get the date the migraine was
recorded
                    //changing the pain intensity from a string to string number
                    graphObj.PainLevel = (float)Convert.ToDouble(
obj.painIntensity.Substring(0, 1));//get pain intensity string number
                    recordsForThatMonth.Add(graphObj);
                }
            }
            return recordsForThatMonth;
        }
    }
}
```

4.5 MigraineTrackingApp.Android

4.5.1 LoginAndRegister.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using System;
using System.Threading.Tasks;
using Android.Gms.Extensions;
using Firebase.Auth;
using MigraineTrackingApp;
using Xamarin.Forms;

[assembly: Dependency(typeof(LoginAndRegister))]
namespace MigraineTrackingApp
{
    public class LoginAndRegister : IAuth
    {
        public async Task<string> LoginWithEmailPassword(string email, string
password)
        {
            try
            {
                //Reference: https://www.lindseybroos.be/2020/03/xamarin-forms-and-
firebase-authentication/

                var user = await
FirebaseAuth.Instance.SignInWithEmailAndPasswordAsync(email, password);
                var token = user.User.Uid;
                string userId = token.ToString();
                return userId;
            }
            catch (FirebaseAuthInvalidUserException e)
            {
                e.PrintStackTrace();
                return "";
            }
            catch (FirebaseAuthInvalidCredentialsException e)
            {
                e.PrintStackTrace();
                return "";
            }
        }

        public async Task<string> SignupWithEmailPassword(string email, string
password)
        {
            try
            {
                var user = await
FirebaseAuth.Instance.CreateUserWithEmailAndPasswordAsync(email,password);
                var token = user.User.Uid;
                string userId = token.ToString();
                return userId;
            }
            catch (FirebaseAuthInvalidUserException e)
```

```
            {
                e.PrintStackTrace();
                return "";
            }
        }
        public void Logout()
        {
            FirebaseAuth.Instance.SignOut();
        }
    }
}
```

4.5.2 MainActivity.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using System;

using Android.App;
using Android.Content.PM;
using Android.Runtime;
using Android.OS;
using Firebase;

namespace MigraineTrackingApp.Droid
{
    [Activity(Label = "migraineHub", Icon = "@mipmap/icon", Theme =
"@style/MainTheme", MainLauncher = true, ConfigurationChanges =
ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode |
ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize )]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            Xamarin.Essentials.Platform.Init(this, savedInstanceState);
            global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
            ZXing.Net.Mobile.Forms.Android.Platform.Init();

            FirebaseApp.InitializeApp(Application.Context);
            LoadApplication(new App());
        }
        public override void OnRequestPermissionsResult(int requestCode, string[]
permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
        {
            Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);



global::ZXing.Net.Mobile.Android.PermissionsHandler.OnRequestPermissionsResult(request
Code, permissions, grantResults);



            base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }
}
```

4.5.3 colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="launcher_background">#FFFFFF</color>
    <color name="colorPrimary">#00905E</color>
    <color name="colorPrimaryDark">#00905E</color>
    <color name="colorAccent">#007361</color>
</resources>
```

4.5.4 styles.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>

	<style name="MainTheme" parent="MainTheme.Base">
		<!-- As of Xamarin.Forms 4.6 the theme has moved into the Forms binary -
->
		<!-- If you want to override anything you can do that here. -->
		<!-- Underneath are a couple of entries to get you started. -->

		<!-- Set theme colors from https://aka.ms/material-colors -->
		<!-- colorPrimary is used for the default action bar background -->
		<item name="colorPrimary">#00905E</item>
		<!-- colorPrimaryDark is used for the status bar -->
	    <item name="colorPrimaryDark">#00905E</item>
		<!-- colorAccent is used as the default value for colorControlActivated
	  which is used to tint widgets -->
		<item name="colorAccent">#007361</item>

		<item
name="android:colorPressedHighlight">@color/ListViewSelected</item>
		<item
name="android:colorLongPressedHighlight">@color/ListViewHighlighted</item>
		<item
name="android:colorFocusedHighlight">@color/ListViewSelected</item>
		<item
name="android:colorActivatedHighlight">@color/ListViewSelected</item>
		<item
name="android:activatedBackgroundIndicator">@color/ListViewSelected</item>


		<item
name="android:datePickerDialogTheme">@style/CustomDatePickerDialog</item>


	</style>
	<color name="ListViewSelected">#68C551</color>
	<color name="ListViewHighlighted">#68C551</color>



	<style name="CustomDatePickerDialog" parent="ThemeOverlay.AppCompat.Dialog">
		<!--header background-->
		<item name="colorAccent">#00AB58</item>
		<!--header textcolor-->
		<item name="android:textColorPrimaryInverse">#FFFFFF</item>
		<!--body background-->
		<item name="android:windowBackground">#FFFFFF</item>
		<!--selected day-->
		<item name="android:colorControlActivated">#00AB58</item>
		<!--days of the month-->
		<item name="android:textColorPrimary">#000000</item>
		<!--days of the week-->
		<item name="android:textColorSecondary">#00AB58</item>
		<!--cancel&ok-->
		<item name="android:textColor">#005248</item>
	</style>

</resources>
```

4.6 MigraineTrackingApp.iOS

4.6.1 LoginAndRegister.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number: C00242743
 * Date: 18/4/2022
 */

using System;
using System.Threading.Tasks;
using Xamarin.Forms;
using Firebase.Auth;
using MigraineTrackingApp.iOS;

[assembly: Dependency(typeof(LoginAndRegister))]

namespace MigraineTrackingApp.iOS
{
    class LoginAndRegister : IAuth
    {
        public async Task<string> LoginWithEmailPassword(string email, string
password)
        {
            try
            {
                var user = await Auth.DefaultInstance.SignInWithPasswordAsync(email,
password);
                return await user.User.GetIdTokenAsync();
            }
            catch (Exception e)
            {
                return "";
            }
        }

        public Task<string> SignupWithEmailPassword(string email, string password)
        {
            throw new NotImplementedException();
        }
        public void Logout()
        {
            Foundation.NSError error;
            Auth.DefaultInstance.SignOut(out error);
        }
    }
}
```

4.6.2 Main.cs

```
/*
 * Student Name: Michelle Bolger
 * Student Number C00242743
 */

using System;
using System.Collections.Generic;
using System.Linq;

using Foundation;
using UIKit;

namespace MigraineTrackingApp.iOS
{
    public class Application
    {
        // This is the main entry point of the application.
        static void Main(string[] args)
        {
            // if you want to use a different Application Delegate class from
"AppDelegate"
            // you can specify it here.
            UIApplication.Main(args, null, "AppDelegate");
        }
    }
}
```