



INSTITUTE *of*
TECHNOLOGY

CARLOW

Institiúid Teicneolaíochta Cheatharlach

Application Green Quake



Student Name: Peter Lucan

Student Number: C00228946

Supervisor: Chris Meudec

Github Link:

<https://github.com/PeterX12/Application-Green-Quake.git>

APK Github Link:

<https://github.com/PeterX12/Application-Green-Quake/blob/master/ApplicationGreenQuake.apk>

APK Google Drive Link:

https://drive.google.com/file/d/1MIPChFA_Z_XOzcy3yi_Q1Gt6JyNJcC1e/view?usp=sharing

Abstract

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

Table of Contents

1 Introduction	3
2 Documentation	3
3 Installation	3
4 Application Screenshots	4
4.1 Login Screen	4
4.2 Login Screen With Error Messages	5
4.3 Sign Up Screen	5
4.4 Sign Up Screen With Error Messages	6
4.5 Forgot Password Screen	6
4.6 Forgot Password Screen With Error Messages	7
4.7 Main Menu Screen	7
4.8 Categories Screen	8
4.9 Food and Drink Subcategories Screen	8
4.10 H2O Action Screen	9
4.11 Refill Stations Screen	9
4.12 Leaderboard Screen	10
4.13 Leaderboard Screen When The Filter Button Is Tapped	10
4.14 Leaderboard Pop Up Screen When A Profile Is Tapped On The Leaderboard	11
4.15 Filtered Leaderboard Screen	11
4.16 Profile Screen	12
4.17 Popup Screen After The Profile Image Is Tapped	12
4.18 The Trophies Screen	13
4.19 The Achievements Screen	13
4.20 The BadgesScreen	14
4.21 The Badge Popup Screen	14
4.22 The Live Avatar Mosaics on Lvl 56	15
4.23 The Live Avatar Mosaics on Lvl 57	15
4.24 The Live Avatar Mosaics on Lvl 58	16
4.25 The Live Avatar Mosaics on Lvl 59	16
4.26 The Live Avatar Mosaics on Lvl 60	17
5 Database	17
5.1 Firebase Database	18
5.2 Firebase Storage	21
5.3 Firebase Authentication	22
6 Code Listings	22

1 Introduction

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

2 Documentation

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

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

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

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

3 Installation

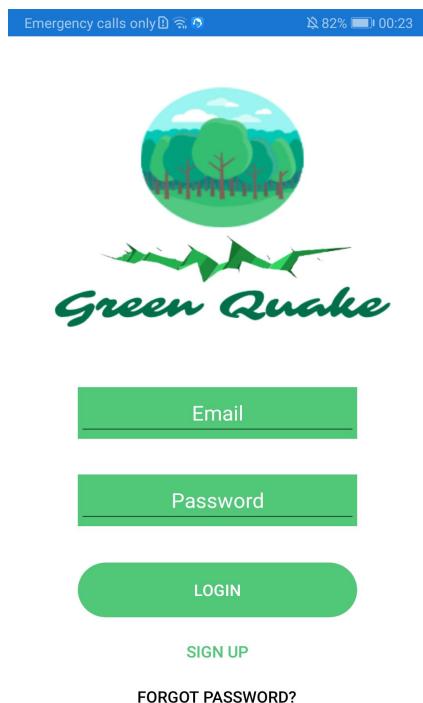
The application can be installed in two ways. The first way and the more complicated way is to clone the repository and open the solution using Visual Studio, connecting your device using a USB and selecting it from the drop down and clicking the play button.

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

4 Application Screenshots

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

4.1 Login Screen

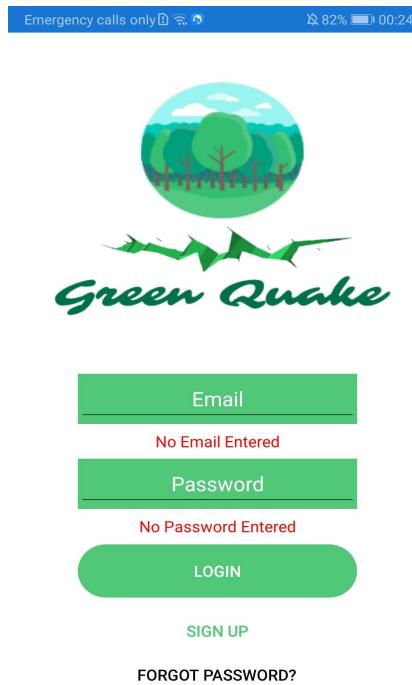


As you can see this is the **login** screen. This will be the opening screen for the application when the user is not signed in. A user can enter their email and password and click the login button to attempt to sign in or they can tap the Sign up option to navigate to the sign up page or they can tap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

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

Figure 1: Login Screen.

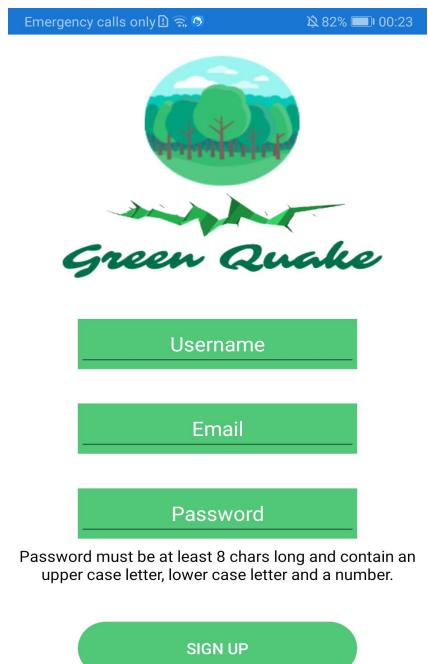
4.2 Login Screen With Error Messages



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

Figure 2: Login Screen With Error Messages.

4.3 Sign Up Screen



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

Figure 3: Sign Up Screen.

4.4 Sign Up Screen With Error Messages

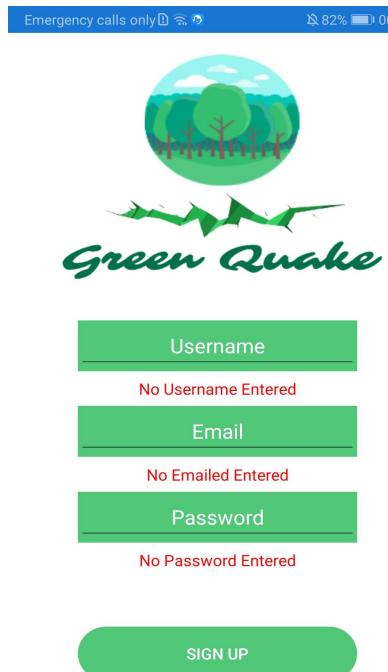


Figure 4: Sign Up Screen With Error Messages.

4.5 Forgot Password Screen

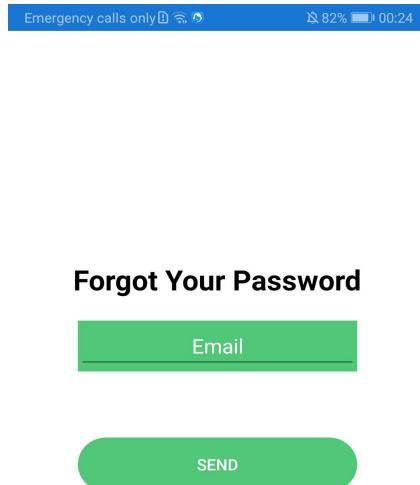


Figure 5: Forgot Password Screen.

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

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

4.6 Forgot Password Screen With Error Messages

Emergency calls only 82% 00:24

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

Forgot Your Password

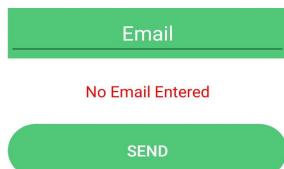
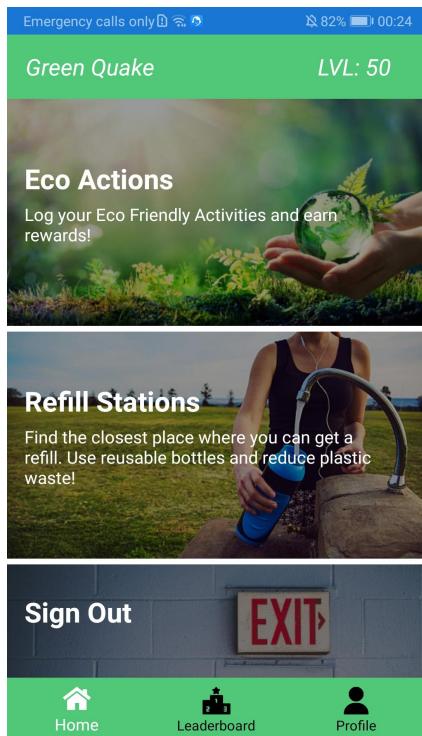


Figure 6: Forgot Password Screen With Error Messages.

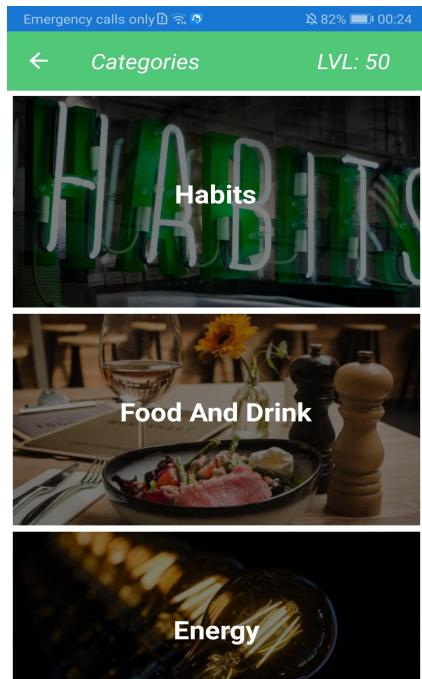
4.7 Main Menu Screen



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

Figure 7: Main Menu Screen.

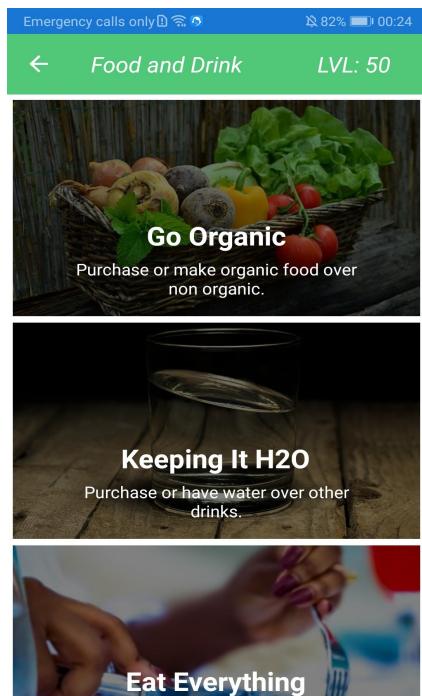
4.8 Categories Screen



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

Figure 8: Categories Screen.

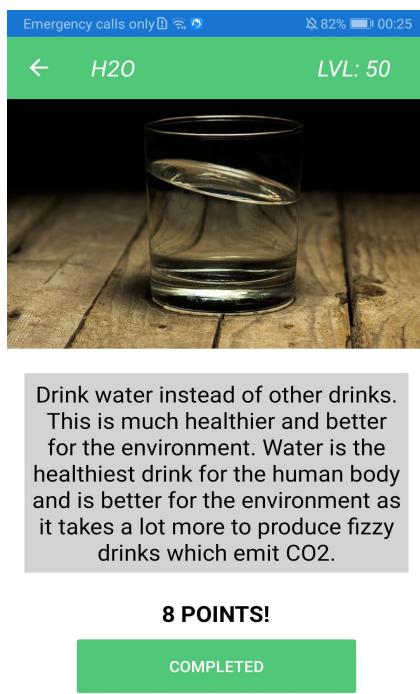
4.9 Food and Drink Subcategories Screen



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

Figure 9: Food and Drink Subcategories Screen.

4.10 H2O Action Screen



This is the **H2O** action screen. This screen provides some details and interesting information about this particular action. The amount of points that it is worth is displayed below the text. The user can press the Completed button to log the action. Once this is pressed either the points get posted into the database and an alert box with the successful message is displayed to the user before redirecting them to the main menu or one of two error message alert boxes pop up.

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

Figure 10: H2O Action Screen.

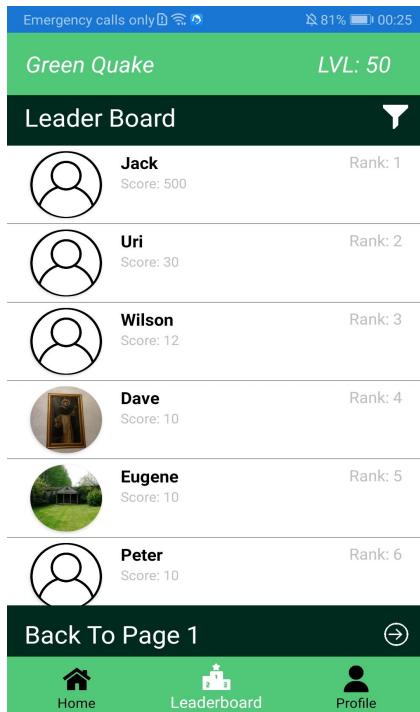
4.11 Refill Stations Screen



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

Figure 11: Refill Stations Screen.

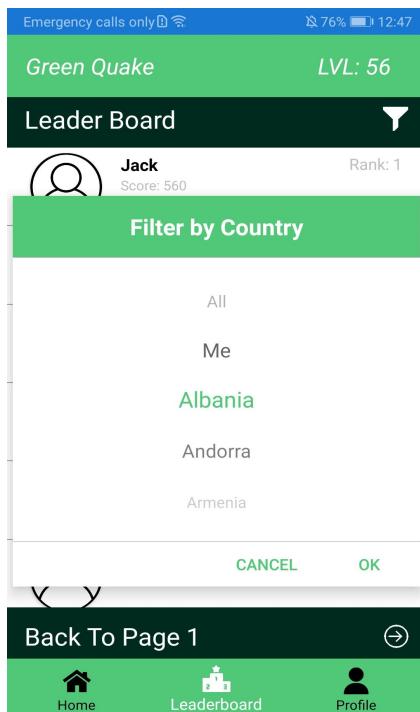
4.12 Leaderboard Screen



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

Figure 12: Leaderboard Screen.

4.13 Leaderboard Screen When The Filter Button Is Tapped



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

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

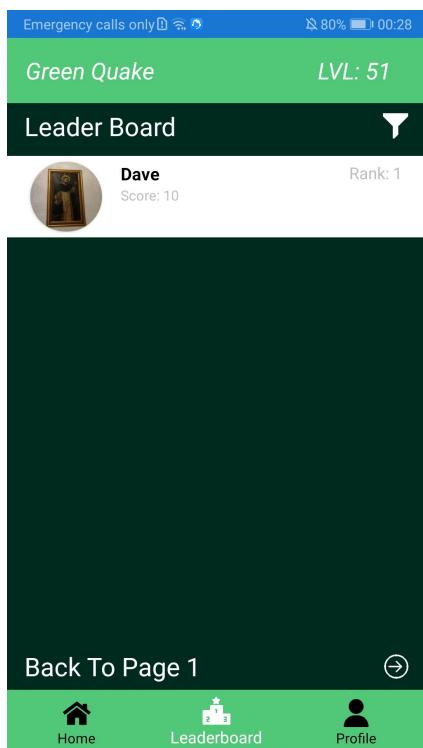
4.14 Leaderboard Pop Up Screen When A Profile Is Tapped On The Leaderboard



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

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

4.15 Filtered Leaderboard Screen



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

Figure 22: Filtered Leaderboard Screen.

4.16 Profile Screen

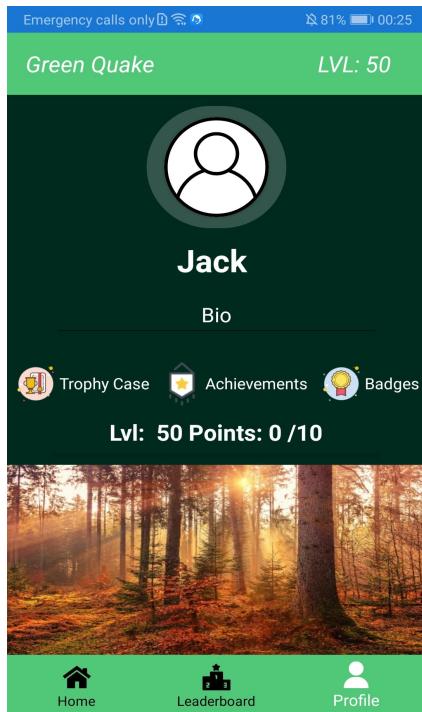
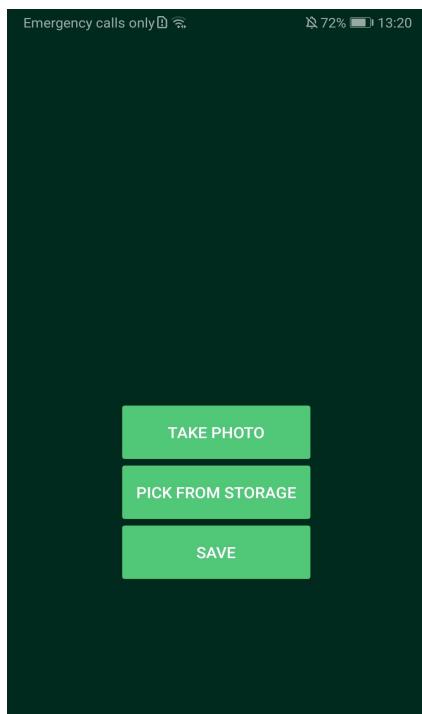


Figure 16: Profile Screen.

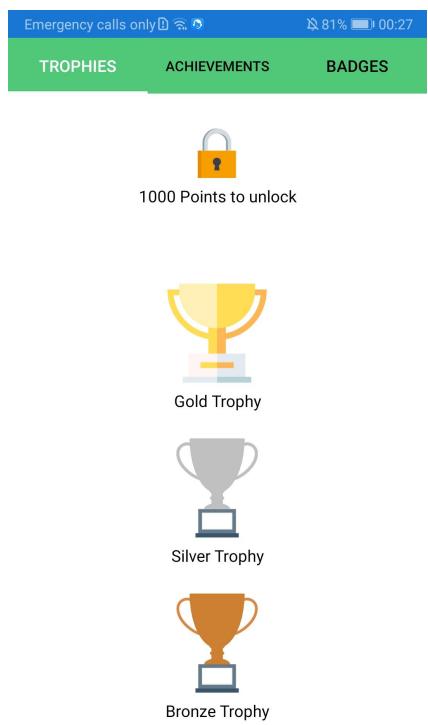
4.17 Popup Screen After The Profile Image Is Tapped



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

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

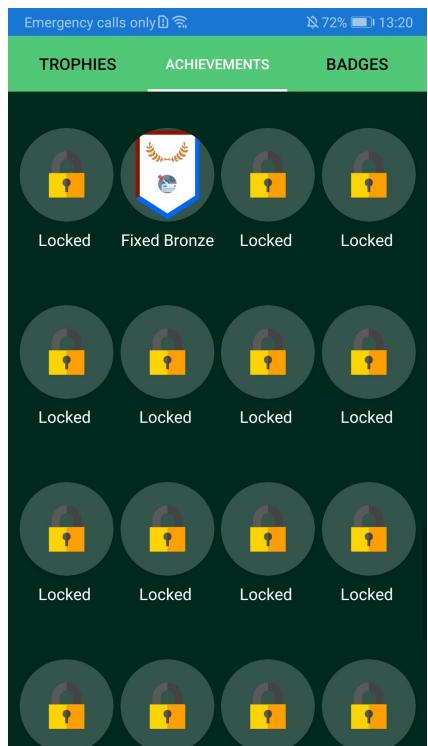
4.18 The Trophies Screen



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

Figure 18: The Trophies Screen.

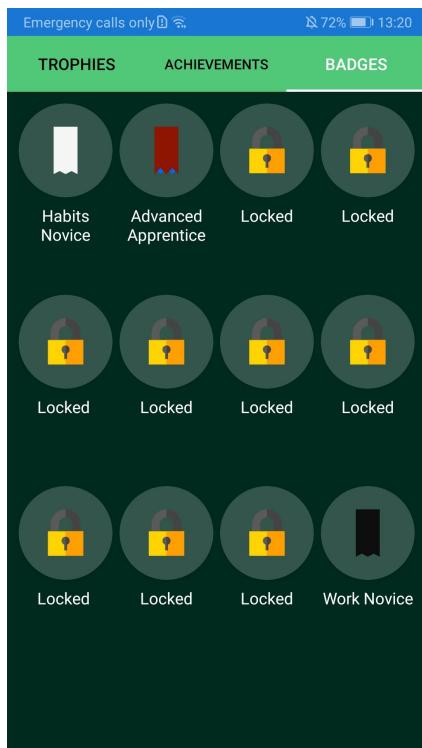
4.19 The Achievements Screen



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

Figure 19: The Achievements Screen.

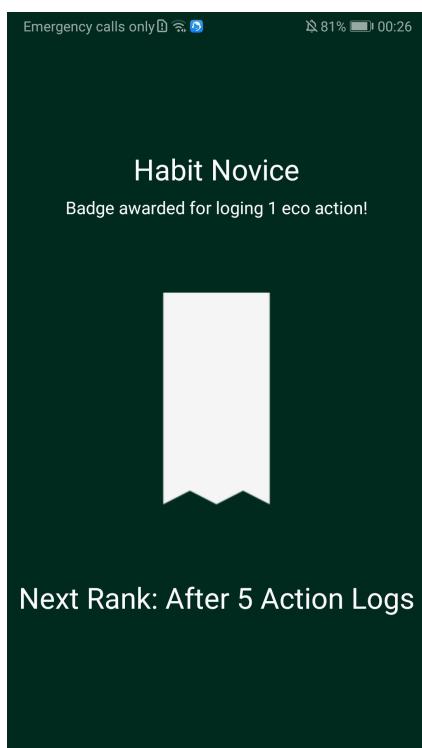
4.20 The Badges Screen



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

Figure 20: The Badges Screen.

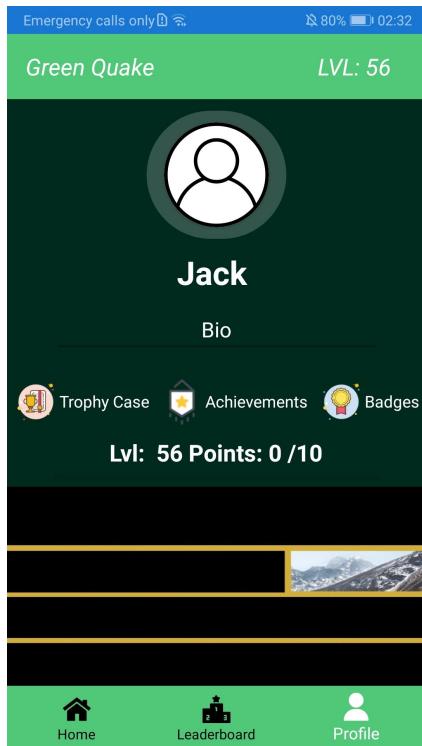
4.21 The Badge Popup Screen



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

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

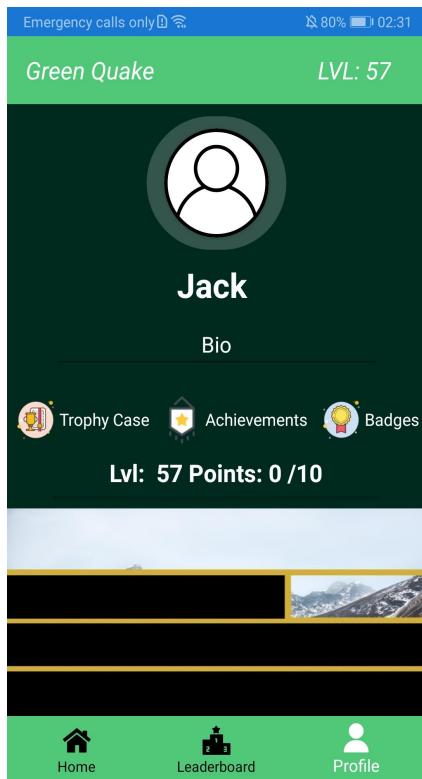
4.22 The Live Avatar Mosaics on Lvl 56



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

Figure 22: The Live Avatar Mosaics on Lvl 56.

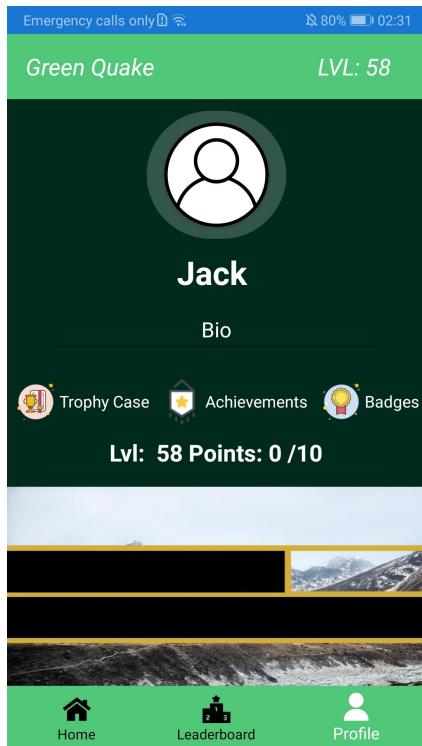
4.23 The Live Avatar Mosaics on Lvl 57



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

Figure 23: The Live Avatar Mosaics on Lvl 57.

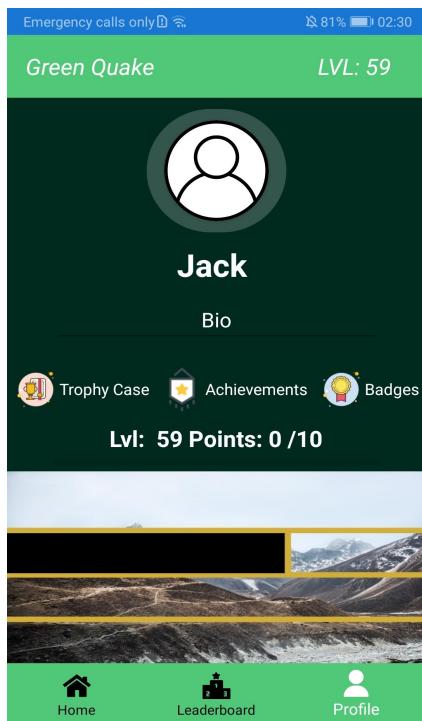
4.24 The Live Avatar Mosaics on Lvl 58



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

Figure 24: The Live Avatar Mosaics on Lvl 58.

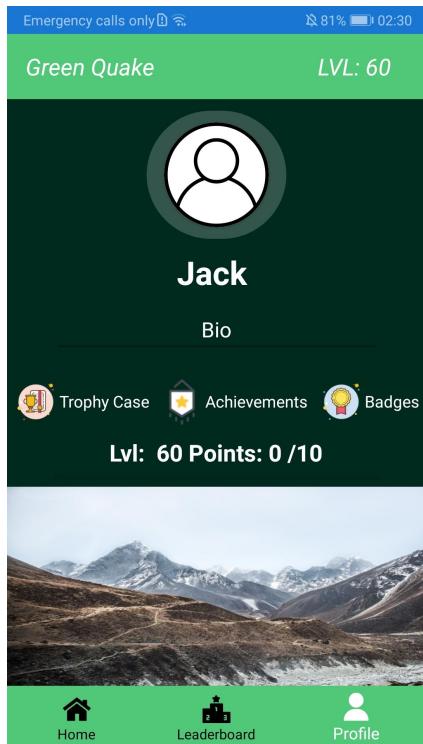
4.25 The Live Avatar Mosaics on Lvl 59



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

Figure 25: The Live Avatar Mosaics on Lvl 59.

4.26 The Live Avatar Mosaics on Lvl 60



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

Figure 26: The Live Avatar Mosaics on Lvl 60.

5 Database

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

5.1 Firebase Database

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

```
-|FoodAndDrinkPoints
  -|UID
    -|eatAllCount
    -|foodDeliverCount
    -|noMeatCount
    -|numberOfLogs
    -|organicCount
    -|ownCoffeeCount
    -|points
    -|reCoffeeMugCount
    -|saveLeftOversCount
    -|steelStrawCount
    -|username
    -|waterOverFizzyCount

-|HabitsPoints
  -|UID
    -|brushingCount
    -|fullWasherCount
    -|matchesCount
    -|numberOfLogs
    -|offLigtsCount
    -|points
    -|showerCount
    -|timedShowerCount
    -|username
```

-|Points

-|UID

-|points

-|username

-|SecuritySchecks

-|UID

-|counter

-|date

-|time

-|Station

-|ID

-|description

-|label

-|latitude

-|longitude

-|Travel

-|UID

-|carpoolCount

-|cycleCount

-|ecoCarCount

-|numberOfLogs

-|points

-|transportCount

-|username

-|walkCount

```
-|WastePoints
  -|UID
    -|billsCount
    -|bioBinBagsCount
    -|compostCount
    -|numberOfLogs
    -|points
    -|recyclingBinCount
    -|setUpRecyclingBinCount
    -|username

-|WorkPoints
  -|UID
    -|numberOfLogs
    -|offElectronicsCount
    -|paperCount
    -|points
    -|remoteWorkCount
    -|username

-|usernames
  -|username
    -|Uid

-|Users
  -|UID
    -|bio
    -|nation
    -|username
```

5.2 Firebase Storage

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

`-lgs://application-green-quake.appspot.com`

-|UID

-|filename

5.3 Firebase Authentication

The Firebase Authentication Database is shown below.

Identifier	Providers	Created	Signed in	User UID ↑
r@r.com	✉	25 Apr 2021	25 Apr 2021	1z5PxJeIGaoOUMICnsBweDFPzU2
i@i.com	✉	25 Apr 2021	25 Apr 2021	IXDkYbSKgyQ1rE2RpMBvW83jrFI1
d@d.com	✉	25 Apr 2021	25 Apr 2021	JTgqfrU8t4aMxKluZQ6ohtdZn3I3
e@e.com	✉	25 Apr 2021	25 Apr 2021	M3VwYe6T8wP1InABBKmGqq4o0...
u@u.com	✉	25 Apr 2021	25 Apr 2021	UPVjIHV3AT7Ip7YeDVFGZ41ISe2
o@o.com	✉	25 Apr 2021	25 Apr 2021	XEXN5KzRKtToLKutUxaUifngl2u2
q@q.com	✉	25 Apr 2021	25 Apr 2021	YKcG1vRnX3c4ZoUTEZMEEQ4ZYf...
w@w.com	✉	25 Apr 2021	25 Apr 2021	ILG6ybBODPb6dfPotSxdEoExH5J2
jack@jack.com	✉	25 Apr 2021	29 Apr 2021	jSzcORiL1cdLSdikiDyJ8qrLcAU2
bo@bo.com	✉	27 Apr 2021	27 Apr 2021	kFxIXXvPe8Mj42Bi9oJOI2Y6o5K2
peter@peter.com	✉	25 Apr 2021	25 Apr 2021	ozaGzbHeiVWMtPffzHE6II2Jy8g1
y@y.com	✉	25 Apr 2021	27 Apr 2021	rE5eSerKG5fvi2CYVglhT8fGGUg2

6 Code Listings

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

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

```
1 /*! \class The AppConstants Model Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the AppConstants Model Class in contains the constants that are used
8  * throughout this application.
9  */
10 {
11     public class AppConstants
12     {
13         //Int Constants
14         public const int twoPoints = 2;
15         public const int fourPoints = 4;
16         public const int sixPoints = 6;
17         public const int eightPoints = 8;
18         public const int tenPoints = 10;
19
20         //String Constants
21         public const string googleMapsApiKey = "AIzaSyDf7Bq7gjei8Sp1AS_SWeapWyHe2rJtLmw";
22         public const string twoPointsMsg = "2 Points Point Have been added";
23         public const string fourPointsMsg = "4 Points Point Have been added";
24         public const string sixPointsMsg = "6 Points Point Have been added";
25         public const string eightPointsMsg = "8 Points Point Have been added";
26         public const string tenPointsMsg = "10 Points Point Have been added";
27     }
28 }
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
1 /*! \class The AdvancedPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the AdvancedPointsUpdate ViewModel Class. It updates the data for
8  * the Advanced Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 * this data to back firebase.
11 *
12 */
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class AdvancedPointsUpdate
22     {
23         int points2 = 0;
24         int number0fLogs2 = 0;
25         int fixCount2 = 0;
26         string username = "";
27
28         IAuth auth;
29         /** This function updates the points in the Advanced category by ten points. It
30 also increments the number of logs logged in the Advanced
31      * category by one and increments the number of times this particular action was
32 logged by one and sends this data to Firebase.
33 */
34         public async void FixPoints()
35         {
36             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
37 quake-default-rtdb.firebaseio.com/");
38             auth = DependencyService.Get<IAuth>();
39             try
40             {
41                 username = (await firebaseClient
42                     .Child("users")
43                     .Child(auth.GetUid())
44                     .OnceSingleAsync<Users>()).username;
45
46                 points2 = (await firebaseClient
47                     .Child("AdvancedPoints")
48                     .Child(auth.GetUid())
49                     .OnceSingleAsync<AdvancedPoints>()).points;
50
51                 points2 = points2 + AppConstants.tenPoints;
52
53                 number0fLogs2 = (await firebaseClient
54                     .Child("AdvancedPoints")
55                     .Child(auth.GetUid())
56                     .OnceSingleAsync<AdvancedPoints>()).number0fLogs;
```

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

```
1 /*! \class The CommunityPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the CommunityPointsUpdate ViewModel Class. It updates the data for
8  * the Community Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 * this data to back firebase.
11 *
12 */
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class CommunityPointsUpdate
22     {
23         int points2 = 0;
24         int createGroupCount2 = 0;
25         int communityCount2 = 0;
26         int donateCount2 = 0;
27         int groupCount2 = 0;
28         int shareCount2 = 0;
29         int awarenessCount2 = 0;
30         int numberofLogs2 = 0;
31
32         string username = "";
33
34         IAuth auth;
35         /** This function updates the points in the Community category by ten points. It
36         also increments the number of logs logged in the Community
37         * category by one and increments the number of times this particular action was
38         logged by one and sends this data to Firebase.
39         */
40         public async void CreateGroupPoints()
41         {
42
43             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
44             quake-default-rtdb.firebaseio.com/");
45             auth = DependencyService.Get<IAuth>();
46
47             try
48             {
49                 username = (await firebaseClient
50                     .Child("users")
51                     .Child(auth.GetUid())
52                     .OnceSingleAsync<Users>()).username;
53
54                 points2 = (await firebaseClient
55                     .Child("CommunityPoints")
56                     .Child(auth.GetUid())
57                     .OnceSingleAsync<CommunityPoints>()).points;
58
59                 points2 = points2 + AppConstants.tenPoints;
60
61                 numberofLogs2 = (await firebaseClient
```

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

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

```
175     communityCount2 = (await firebaseClient
176         .Child("CommunityPoints")
177         .Child(auth.GetUid())
178         .OnceSingleAsync<CommunityPoints>()).communityCount;
179
180     communityCount2++;
181
182     donateCount2 = (await firebaseClient
183         .Child("CommunityPoints")
184         .Child(auth.GetUid())
185         .OnceSingleAsync<CommunityPoints>()).donateCount;
186
187     groupCount2 = (await firebaseClient
188         .Child("CommunityPoints")
189         .Child(auth.GetUid())
190         .OnceSingleAsync<CommunityPoints>()).groupCount;
191
192     shareCount2 = (await firebaseClient
193         .Child("CommunityPoints")
194         .Child(auth.GetUid())
195         .OnceSingleAsync<CommunityPoints>()).shareCount;
196
197     awarenessCount2 = (await firebaseClient
198         .Child("CommunityPoints")
199         .Child(auth.GetUid())
200         .OnceSingleAsync<CommunityPoints>()).awarenessCount;
201
202     await firebaseClient
203         .Child("CommunityPoints")
204         .Child(auth.GetUid())
205         .PutAsync(new CommunityPoints()
206     {
207         username = username,
208         points = points2,
209         numberLogs = numberLogs2,
210         communityCount = communityCount2,
211         createGroupCount = createGroupCount2,
212         donateCount = donateCount2,
213         groupCount = groupCount2,
214         shareCount = shareCount2,
215         awarenessCount = awarenessCount2,
216
217     });
218 }
219 catch (FirebaseException)
220 {
221     username = (await firebaseClient
222         .Child("users")
223         .Child(auth.GetUid())
224         .OnceSingleAsync<Users>()).username;
225
226     points2 = AppConstants.tenPoints;
227     await firebaseClient
228         .Child("CommunityPoints")
229         .Child(auth.GetUid())
230         .PutAsync(new CommunityPoints() { username = username, points = points2,
numberLogs = 1, communityCount = 1}); ;
231
232 }
233 catch (NullReferenceException)
234 {
```

```
235     username = (await firebaseClient
236         .Child("users")
237         .Child(auth.GetUid())
238         .OnceSingleAsync<Users>()).username;
239
240     points2 = AppConstants.tenPoints;
241     await firebaseClient
242         .Child("CommunityPoints")
243         .Child(auth.GetUid())
244         .PutAsync(new CommunityPoints() { username = username, points = points2,
245     numberOfLogs = 1, communityCount = 1 });
246 }
247 /**
248 * This function updates the points in the Community category by ten points. It
249 * also increments the number of logs logged in the Community
250 * category by one and increments the number of times this particular action was
251 * logged by one and sends this data to Firebase.
252 */
253 public async void DonatePoints()
254 {
255
256     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
257     quake-default-rtdb.firebaseio.com/");
258     auth = DependencyService.Get<IAuth>();
259
260     try
261     {
262         username = (await firebaseClient
263             .Child("users")
264             .Child(auth.GetUid())
265             .OnceSingleAsync<Users>()).username;
266
267         points2 = (await firebaseClient
268             .Child("CommunityPoints")
269             .Child(auth.GetUid())
270             .OnceSingleAsync<CommunityPoints>()).points;
271
272         points2 = points2 + AppConstants.tenPoints;
273
274         numberOfLogs2 = (await firebaseClient
275             .Child("CommunityPoints")
276             .Child(auth.GetUid())
277             .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
278
279         numberOfLogs2++;
280
281         createGroupCount2 = (await firebaseClient
282             .Child("CommunityPoints")
283             .Child(auth.GetUid())
284             .OnceSingleAsync<CommunityPoints>()).createGroupCount;
285
286         communityCount2 = (await firebaseClient
287             .Child("CommunityPoints")
288             .Child(auth.GetUid())
289             .OnceSingleAsync<CommunityPoints>()).communityCount;
290
291         donateCount2 = (await firebaseClient
292             .Child("CommunityPoints")
293             .Child(auth.GetUid())
294             .OnceSingleAsync<CommunityPoints>()).donateCount;
295
296         donateCount2++;
297     }
298 }
```

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

```
353     }
354     /** This function updates the points in the Community category by eight points. It
355      also increments the number of logs logged in the Community
356      * category by one and increments the number of times this particular action was
357      logged by one and sends this data to Firebase.
358      */
359     public async void GroupPoints()
360     {
361
362         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
363         quake-default-rtdb.firebaseio.com/");
364         auth = DependencyService.Get<IAuth>();
365
366         try
367         {
368             username = (await firebaseClient
369                         .Child("users")
370                         .Child(auth.GetUid())
371                         .OnceSingleAsync<Users>()).username;
372
373             points2 = (await firebaseClient
374                         .Child("CommunityPoints")
375                         .Child(auth.GetUid())
376                         .OnceSingleAsync<CommunityPoints>()).points;
377
378             points2 = points2 + AppConstants.eightPoints;
379
380             numberOfLogs2 = (await firebaseClient
381                             .Child("CommunityPoints")
382                             .Child(auth.GetUid())
383                             .OnceSingleAsync<CommunityPoints>()).numberOfLogs;
384
385             numberOfLogs2++;
386
387             createGroupCount2 = (await firebaseClient
388                             .Child("CommunityPoints")
389                             .Child(auth.GetUid())
390                             .OnceSingleAsync<CommunityPoints>()).createGroupCount;
391
392             communityCount2 = (await firebaseClient
393                             .Child("CommunityPoints")
394                             .Child(auth.GetUid())
395                             .OnceSingleAsync<CommunityPoints>()).communityCount;
396
397             donateCount2 = (await firebaseClient
398                             .Child("CommunityPoints")
399                             .Child(auth.GetUid())
400                             .OnceSingleAsync<CommunityPoints>()).donateCount;
401
402             groupCount2 = (await firebaseClient
403                             .Child("CommunityPoints")
404                             .Child(auth.GetUid())
405                             .OnceSingleAsync<CommunityPoints>()).groupCount;
406
407             groupCount2++;
408
409             shareCount2 = (await firebaseClient
410                             .Child("CommunityPoints")
411                             .Child(auth.GetUid())
412                             .OnceSingleAsync<CommunityPoints>()).shareCount;
413
414             awarenessCount2 = (await firebaseClient
```

```
412         .Child("CommunityPoints")
413         .Child(auth.GetUid())
414         .OnceSingleAsync<CommunityPoints>()).awarenessCount;
415
416         await firebaseClient
417         .Child("CommunityPoints")
418         .Child(auth.GetUid())
419         .PutAsync(new CommunityPoints()
420         {
421             username = username,
422             points = points2,
423             numberLogs = numberLogs2,
424             communityCount = communityCount2,
425             createGroupCount = createGroupCount2,
426             donateCount = donateCount2,
427             groupCount = groupCount2,
428             shareCount = shareCount2,
429             awarenessCount = awarenessCount2,
430
431         });
432     }
433     catch (FirebaseException)
434     {
435         username = (await firebaseClient
436         .Child("users")
437         .Child(auth.GetUid())
438         .OnceSingleAsync<Users>()).username;
439
440         points2 = AppConstants.eightPoints;
441         await firebaseClient
442         .Child("CommunityPoints")
443         .Child(auth.GetUid())
444         .PutAsync(new CommunityPoints() { username = username, points = points2,
445         numberLogs = 1, groupCount = 1 });
446     }
447     catch (NullReferenceException)
448     {
449         username = (await firebaseClient
450         .Child("users")
451         .Child(auth.GetUid())
452         .OnceSingleAsync<Users>()).username;
453
454         points2 = AppConstants.eightPoints;
455         await firebaseClient
456         .Child("CommunityPoints")
457         .Child(auth.GetUid())
458         .PutAsync(new CommunityPoints() { username = username, points = points2,
459         numberLogs = 1, groupCount = 1 });
460     }
461     /** This function updates the points in the Community category by eight points. It
462     also increments the number of logs logged in the Community
463     * category by one and increments the number of times this particular action was
464     logged by one and sends this data to Firebase.
465     */
466     public async void SharePoints()
467     {
468
469         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
470         quake-default-rtdb.firebaseio.com/");
471         auth = DependencyService.Get<IAuth>();
```

```
469
470     try
471     {
472         username = (await firebaseClient
473             .Child("users")
474             .Child(auth.GetUid())
475             .OnceSingleAsync<Users>()).username;
476
477         points2 = (await firebaseClient
478             .Child("CommunityPoints")
479             .Child(auth.GetUid())
480             .OnceSingleAsync<CommunityPoints>()).points;
481
482         points2 = points2 + AppConstants.eightPoints;
483
484         numberofLogs2 = (await firebaseClient
485             .Child("CommunityPoints")
486             .Child(auth.GetUid())
487             .OnceSingleAsync<CommunityPoints>()).numberofLogs;
488
489         numberofLogs2++;
490
491         createGroupCount2 = (await firebaseClient
492             .Child("CommunityPoints")
493             .Child(auth.GetUid())
494             .OnceSingleAsync<CommunityPoints>()).createGroupCount;
495
496         communityCount2 = (await firebaseClient
497             .Child("CommunityPoints")
498             .Child(auth.GetUid())
499             .OnceSingleAsync<CommunityPoints>()).communityCount;
500
501         donateCount2 = (await firebaseClient
502             .Child("CommunityPoints")
503             .Child(auth.GetUid())
504             .OnceSingleAsync<CommunityPoints>()).donateCount;
505
506         groupCount2 = (await firebaseClient
507             .Child("CommunityPoints")
508             .Child(auth.GetUid())
509             .OnceSingleAsync<CommunityPoints>()).groupCount;
510
511         shareCount2 = (await firebaseClient
512             .Child("CommunityPoints")
513             .Child(auth.GetUid())
514             .OnceSingleAsync<CommunityPoints>()).shareCount;
515
516         shareCount2++;
517
518         awarenessCount2 = (await firebaseClient
519             .Child("CommunityPoints")
520             .Child(auth.GetUid())
521             .OnceSingleAsync<CommunityPoints>()).awarenessCount;
522
523         await firebaseClient
524             .Child("CommunityPoints")
525             .Child(auth.GetUid())
526             .PutAsync(new CommunityPoints()
527             {
528                 username = username,
529                 points = points2,
```

```
530             number0fLogs = number0fLogs2,
531             communityCount = communityCount2,
532             createGroupCount = createGroupCount2,
533             donateCount = donateCount2,
534             groupCount = groupCount2,
535             shareCount = shareCount2,
536             awarenessCount = awarenessCount2,
537
538         });
539     }
540     catch (FirebaseException)
541     {
542         username = (await firebaseClient
543             .Child("users")
544             .Child(auth.GetUid())
545             .OnceSingleAsync<Users>()).username;
546
547         points2 = AppConstants.eightPoints;
548         await firebaseClient
549             .Child("CommunityPoints")
550             .Child(auth.GetUid())
551             .PutAsync(new CommunityPoints() { username = username, points = points2,
552             number0fLogs = 1, shareCount = 1 });
553     }
554     catch (NullReferenceException)
555     {
556         username = (await firebaseClient
557             .Child("users")
558             .Child(auth.GetUid())
559             .OnceSingleAsync<Users>()).username;
560
561         points2 = AppConstants.eightPoints;
562         await firebaseClient
563             .Child("CommunityPoints")
564             .Child(auth.GetUid())
565             .PutAsync(new CommunityPoints() { username = username, points = points2,
566             number0fLogs = 1, shareCount = 1 });
567     }
568     /** This function updates the points in the Community category by eight points. It
569     also increments the number of logs logged in the Community
570     * category by one and increments the number of times this particular action was
571     * logged by one and sends this data to Firebase.
572     */
573     public async void awarenessPoints()
574     {
575
576         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
577         quake-default-rtdb.firebaseio.com/");
578         auth = DependencyService.Get<IAuth>();
579
580         try
581         {
582             username = (await firebaseClient
583                 .Child("users")
584                 .Child(auth.GetUid())
585                 .OnceSingleAsync<Users>()).username;
586
587             points2 = (await firebaseClient
588                 .Child("CommunityPoints")
589                 .Child(auth.GetUid()))
590
591             .PutAsync(new CommunityPoints() { username = username, points = points2,
592             number0fLogs = 1, shareCount = 1 });
593         }
594         catch (NullReferenceException)
595         {
596             username = (await firebaseClient
597                 .Child("users")
598                 .Child(auth.GetUid())
599                 .OnceSingleAsync<Users>()).username;
600
601             points2 = (await firebaseClient
602                 .Child("CommunityPoints")
603                 .Child(auth.GetUid()))
604
605             .PutAsync(new CommunityPoints() { username = username, points = points2,
606             number0fLogs = 1, shareCount = 1 });
607         }
608     }
609 }
```

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

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

```
1 /*! \class The EnergyPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the EnergyPointsUpdate ViewModel Class. It updates the data for
8  * the Energy Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then
10 sending this data to back firebase.
11 *
12 */
13
14 using Application_Green_Quake.Models;
15 using Firebase.Database;
16 using Firebase.Database.Query;
17 using System;
18 using Xamarin.Forms;
19
20 namespace Application_Green_Quake.ViewModels
21 {
22     class EnergyPointsUpdate
23     {
24         int points2 = 0;
25         int numberLogs2 = 0;
26         int hangDry2 = 0;
27         int fullDryerCount2 = 0;
28         int draftSealCount2 = 0;
29         int ductSealCount2 = 0;
30         int efficientThermostatCount2 = 0;
31         int fridgeCount2 = 0;
32         int fullMachineCount2 = 0;
33         int insulateWaterCount2 = 0;
34         int isolateHomeCount2 = 0;
35         int ledLightBulbCount2 = 0;
36         int microwaveCount2 = 0;
37         int solarPanelCount2 = 0;
38         int offSocketCount2 = 0;
39         int reBatteriesCount2 = 0;
40
41         string username = "";
42
43         IAuth auth;
44         /** This function updates the points in the Energy category by ten points. It also
45          increments the number of logs logged in the Energy
46          * category by one and increments the number of times this particular action was
47          logged by one and sends this data to Firebase.
48
49
50         */
51         public async void HangDryPoints()
52         {
53
54             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
55             quake-default-rtdb.firebaseio.com/");
56             auth = DependencyService.Get<IAuth>();

57             try
58             {
59                 username = (await firebaseClient
60                     .Child("users")
61                     .Child(auth.GetUserId())
62                     .OnceSingleAsync<Users>()).username;
```

```
57     points2 = (await firebaseClient
58         .Child("EnergyPoints")
59         .Child(auth.GetUid())
60         .OnceSingleAsync<EnergyPoints>()).points;
61
62     points2 = points2 + AppConstants.tenPoints;
63
64     numberofLogs2 = (await firebaseClient
65         .Child("EnergyPoints")
66         .Child(auth.GetUid())
67         .OnceSingleAsync<EnergyPoints>()).numberofLogs;
68
69     numberofLogs2++;
70
71     hangDry2 = (await firebaseClient
72         .Child("EnergyPoints")
73         .Child(auth.GetUid())
74         .OnceSingleAsync<EnergyPoints>()).hangDryCount;
75
76     hangDry2++;
77
78     draftSealCount2 = (await firebaseClient
79         .Child("EnergyPoints")
80         .Child(auth.GetUid())
81         .OnceSingleAsync<EnergyPoints>()).draftSealCount;
82
83     ductSealCount2 = (await firebaseClient
84         .Child("EnergyPoints")
85         .Child(auth.GetUid())
86         .OnceSingleAsync<EnergyPoints>()).ductSealCount;
87
88     efficientThermostatCount2 = (await firebaseClient
89         .Child("EnergyPoints")
90         .Child(auth.GetUid())
91         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
92
93     fridgeCount2 = (await firebaseClient
94         .Child("EnergyPoints")
95         .Child(auth.GetUid())
96         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
97
98     fullDryerCount2 = (await firebaseClient
99         .Child("EnergyPoints")
100        .Child(auth.GetUid())
101        .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
102
103    fullMachineCount2 = (await firebaseClient
104        .Child("EnergyPoints")
105        .Child(auth.GetUid())
106        .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
107
108    insulateWaterCount2 = (await firebaseClient
109        .Child("EnergyPoints")
110        .Child(auth.GetUid())
111        .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
112
113    ledLightBulbCount2 = (await firebaseClient
114        .Child("EnergyPoints")
115        .Child(auth.GetUid())
116        .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
117
```

```
118         microwaveCount2 = (await firebaseClient
119             .Child("EnergyPoints")
120             .Child(auth.GetUserId())
121             .OnceSingleAsync<EnergyPoints>()).microwaveCount;
122
123         offSocketCount2 = (await firebaseClient
124             .Child("EnergyPoints")
125             .Child(auth.GetUserId())
126             .OnceSingleAsync<EnergyPoints>()).offSocketCount;
127
128         reBatteriesCount2 = (await firebaseClient
129             .Child("EnergyPoints")
130             .Child(auth.GetUserId())
131             .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
132
133         solarPanelCount2 = (await firebaseClient
134             .Child("EnergyPoints")
135             .Child(auth.GetUserId())
136             .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
137
138         isolateHomeCount2 = (await firebaseClient
139             .Child("EnergyPoints")
140             .Child(auth.GetUserId())
141             .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
142
143         await firebaseClient
144             .Child("EnergyPoints")
145             .Child(auth.GetUserId())
146             .PutAsync(new EnergyPoints()
147 {
148             username = username,
149             points = points2,
150             numberOfWorkLogs = numberOfWorkLogs2,
151             hangDryCount = hangDry2,
152             draftSealCount = draftSealCount2,
153             ductSealCount = ductSealCount2,
154             efficientThermostatCount = efficientThermostatCount2,
155             fridgeCount = fridgeCount2,
156             fullDryerCount = fullDryerCount2,
157             insulateWaterCount = insulateWaterCount2,
158             isolateHomeCount = isolateHomeCount2,
159             ledLightBulbCount = ledLightBulbCount2,
160             microwaveCount = microwaveCount2,
161             offSocketCount = offSocketCount2,
162             reBatteriesCount = reBatteriesCount2,
163             solarPanelCount = solarPanelCount2,
164             fullMachineCount = fullMachineCount2
165         });
166     }
167     catch (FirebaseException)
168     {
169         username = (await firebaseClient
170             .Child("users")
171             .Child(auth.GetUserId())
172             .OnceSingleAsync<Users>()).username;
173
174         points2 = AppConstants.tenPoints;
175         await firebaseClient
176             .Child("EnergyPoints")
177             .Child(auth.GetUserId())
178             .PutAsync(new EnergyPoints() { username = username, points = points2,
```

```
179     numberofLogs = 1, hangDryCount = 1});;
180 }
181 catch (NullReferenceException)
182 {
183     username = (await firebaseClient
184         .Child("users")
185         .Child(auth.GetUid())
186         .OnceSingleAsync<Users>()).username;
187
188     points2 = AppConstants.tenPoints;
189     await firebaseClient
190         .Child("EnergyPoints")
191         .Child(auth.GetUid())
192         .PutAsync(new EnergyPoints() { username = username, points = points2,
193     numberofLogs = 1, hangDryCount = 1 });
194 }
195 /**
196 * This function updates the points in the Energy category by ten points. It also
197 increments the number of logs logged in the Energy
198 * category by one and increments the number of times this particular action was
199 logged by one and sends this data to Firebase.
200 */
201 public async void DryerFullPoints()
202 {
203
204     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
205     quake-default-rtdb.firebaseio.com/");
206     auth = DependencyService.Get<IAuth>();
207
208     try
209     {
210         username = (await firebaseClient
211             .Child("users")
212             .Child(auth.GetUid())
213             .OnceSingleAsync<Users>()).username;
214
215         points2 = (await firebaseClient
216             .Child("EnergyPoints")
217             .Child(auth.GetUid())
218             .OnceSingleAsync<EnergyPoints>()).points;
219
220         points2 = points2 + AppConstants.tenPoints;
221
222         numberofLogs2 = (await firebaseClient
223             .Child("EnergyPoints")
224             .Child(auth.GetUid())
225             .OnceSingleAsync<EnergyPoints>()).numberofLogs;
226
227         numberofLogs2++;
228
229         hangDry2 = (await firebaseClient
230             .Child("EnergyPoints")
231             .Child(auth.GetUid())
232             .OnceSingleAsync<EnergyPoints>()).hangDryCount;
233
234         draftSealCount2 = (await firebaseClient
235             .Child("EnergyPoints")
236             .Child(auth.GetUid())
237             .OnceSingleAsync<EnergyPoints>()).draftSealCount;
238
239         ductSealCount2 = (await firebaseClient
```

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

```

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

```

```
355     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
356     quake-default-firebase.firebaseio.com/");
357     auth = DependencyService.Get<IAuth>();
358
359     try
360     {
361         username = (await firebaseClient
362             .Child("users")
363             .Child(auth.GetUid())
364             .OnceSingleAsync<Users>()).username;
365
366         points2 = (await firebaseClient
367             .Child("EnergyPoints")
368             .Child(auth.GetUid())
369             .OnceSingleAsync<EnergyPoints>()).points;
370
371         points2 = points2 + AppConstants.eightPoints;
372
373         numberofLogs2 = (await firebaseClient
374             .Child("EnergyPoints")
375             .Child(auth.GetUid())
376             .OnceSingleAsync<EnergyPoints>()).numberofLogs;
377
378         numberofLogs2++;
379
380         hangDry2 = (await firebaseClient
381             .Child("EnergyPoints")
382             .Child(auth.GetUid())
383             .OnceSingleAsync<EnergyPoints>()).hangDryCount;
384
385         draftSealCount2 = (await firebaseClient
386             .Child("EnergyPoints")
387             .Child(auth.GetUid())
388             .OnceSingleAsync<EnergyPoints>()).draftSealCount;
389
390         ductSealCount2 = (await firebaseClient
391             .Child("EnergyPoints")
392             .Child(auth.GetUid())
393             .OnceSingleAsync<EnergyPoints>()).ductSealCount;
394
395         efficientThermostatCount2 = (await firebaseClient
396             .Child("EnergyPoints")
397             .Child(auth.GetUid())
398             .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
399
400         efficientThermostatCount2++;
401
402         fridgeCount2 = (await firebaseClient
403             .Child("EnergyPoints")
404             .Child(auth.GetUid())
405             .OnceSingleAsync<EnergyPoints>()).fridgeCount;
406
407         fullDryerCount2 = (await firebaseClient
408             .Child("EnergyPoints")
409             .Child(auth.GetUid())
410             .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
411
412         fullMachineCount2 = (await firebaseClient
413             .Child("EnergyPoints")
414             .Child(auth.GetUid())
415             .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
```

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

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

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

```
594
595     solarPanelCount2 = (await firebaseClient
596         .Child("EnergyPoints")
597         .Child(auth.GetUid())
598         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
599
600     isolateHomeCount2 = (await firebaseClient
601         .Child("EnergyPoints")
602         .Child(auth.GetUid())
603         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
604
605     await firebaseClient
606         .Child("EnergyPoints")
607         .Child(auth.GetUid())
608         .PutAsync(new EnergyPoints()
609     {
610         username = username,
611         points = points2,
612         numberofLogs = numberofLogs2,
613         hangDryCount = hangDry2,
614         draftSealCount = draftSealCount2,
615         ductSealCount = ductSealCount2,
616         efficientThermostatCount = efficientThermostatCount2,
617         fridgeCount = fridgeCount2,
618         fullDryerCount = fullDryerCount2,
619         insulateWaterCount = insulateWaterCount2,
620         isolateHomeCount = isolateHomeCount2,
621         ledLightBulbCount = ledLightBulbCount2,
622         microwaveCount = microwaveCount2,
623         offSocketCount = offSocketCount2,
624         reBatteriesCount = reBatteriesCount2,
625         solarPanelCount = solarPanelCount2,
626         fullMachineCount = fullMachineCount2
627     });
628 }
629 catch (FirebaseException)
630 {
631     username = (await firebaseClient
632         .Child("users")
633         .Child(auth.GetUid())
634         .OnceSingleAsync<Users>()).username;
635
636     points2 = AppConstants.tenPoints;
637     await firebaseClient
638         .Child("EnergyPoints")
639         .Child(auth.GetUid())
640         .PutAsync(new EnergyPoints() { username = username, points = points2,
641     numberofLogs = 1, insulateWaterCount = 1 });
642 }
643 catch (NullReferenceException)
644 {
645     username = (await firebaseClient
646         .Child("users")
647         .Child(auth.GetUid())
648         .OnceSingleAsync<Users>()).username;
649
650     points2 = AppConstants.tenPoints;
651     await firebaseClient
652         .Child("EnergyPoints")
653         .Child(auth.GetUid())
```

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

```
712     fullDryerCount2 = (await firebaseClient
713         .Child("EnergyPoints")
714         .Child(auth.GetUserId())
715         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
716
717     fullMachineCount2 = (await firebaseClient
718         .Child("EnergyPoints")
719         .Child(auth.GetUserId())
720         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
721
722     insulateWaterCount2 = (await firebaseClient
723         .Child("EnergyPoints")
724         .Child(auth.GetUserId())
725         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
726
727     ledLightBulbCount2 = (await firebaseClient
728         .Child("EnergyPoints")
729         .Child(auth.GetUserId())
730         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
731
732     microwaveCount2 = (await firebaseClient
733         .Child("EnergyPoints")
734         .Child(auth.GetUserId())
735         .OnceSingleAsync<EnergyPoints>()).microwaveCount;
736
737     offSocketCount2 = (await firebaseClient
738         .Child("EnergyPoints")
739         .Child(auth.GetUserId())
740         .OnceSingleAsync<EnergyPoints>()).offSocketCount;
741
742     reBatteriesCount2 = (await firebaseClient
743         .Child("EnergyPoints")
744         .Child(auth.GetUserId())
745         .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
746
747     solarPanelCount2 = (await firebaseClient
748         .Child("EnergyPoints")
749         .Child(auth.GetUserId())
750         .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
751
752     isolateHomeCount2 = (await firebaseClient
753         .Child("EnergyPoints")
754         .Child(auth.GetUserId())
755         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
756
757     isolateHomeCount2++;
758
759     await firebaseClient
760         .Child("EnergyPoints")
761         .Child(auth.GetUserId())
762         .PutAsync(new EnergyPoints()
763     {
764         username = username,
765         points = points2,
766         numberofLogs = numberofLogs2,
767         hangDryCount = hangDry2,
768         draftSealCount = draftSealCount2,
769         ductSealCount = ductSealCount2,
770         efficientThermostatCount = efficientThermostatCount2,
771         fridgeCount = fridgeCount2,
772         fullDryerCount = fullDryerCount2,
```

```

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

```

```
830         .OnceSingleAsync<EnergyPoints>()).points;
831
832         points2 = points2 + AppConstants.tenPoints;
833
834         numberLogs2 = (await firebaseClient
835             .Child("EnergyPoints")
836             .Child(auth.GetUid())
837             .OnceSingleAsync<EnergyPoints>()).numberLogs;
838
839         numberLogs2++;
840
841         hangDry2 = (await firebaseClient
842             .Child("EnergyPoints")
843             .Child(auth.GetUid())
844             .OnceSingleAsync<EnergyPoints>()).hangDryCount;
845
846         draftSealCount2 = (await firebaseClient
847             .Child("EnergyPoints")
848             .Child(auth.GetUid())
849             .OnceSingleAsync<EnergyPoints>()).draftSealCount;
850
851         ductSealCount2 = (await firebaseClient
852             .Child("EnergyPoints")
853             .Child(auth.GetUid())
854             .OnceSingleAsync<EnergyPoints>()).ductSealCount;
855
856         efficientThermostatCount2 = (await firebaseClient
857             .Child("EnergyPoints")
858             .Child(auth.GetUid())
859             .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
860
861         fridgeCount2 = (await firebaseClient
862             .Child("EnergyPoints")
863             .Child(auth.GetUid())
864             .OnceSingleAsync<EnergyPoints>()).fridgeCount;
865
866         fullDryerCount2 = (await firebaseClient
867             .Child("EnergyPoints")
868             .Child(auth.GetUid())
869             .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
870
871         fullMachineCount2 = (await firebaseClient
872             .Child("EnergyPoints")
873             .Child(auth.GetUid())
874             .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
875
876         insulateWaterCount2 = (await firebaseClient
877             .Child("EnergyPoints")
878             .Child(auth.GetUid())
879             .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
880
881         ledLightBulbCount2 = (await firebaseClient
882             .Child("EnergyPoints")
883             .Child(auth.GetUid())
884             .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
885
886         ledLightBulbCount2++;
887
888         microwaveCount2 = (await firebaseClient
889             .Child("EnergyPoints")
890             .Child(auth.GetUid()))
```

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

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

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

```

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

```

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

```
1190         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
1191
1192         microwaveCount2 = (await firebaseClient
1193             .Child("EnergyPoints")
1194             .Child(auth.GetUid())
1195             .OnceSingleAsync<EnergyPoints>()).microwaveCount;
1196
1197         microwaveCount2++;
1198
1199         offSocketCount2 = (await firebaseClient
1200             .Child("EnergyPoints")
1201             .Child(auth.GetUid())
1202             .OnceSingleAsync<EnergyPoints>()).offSocketCount;
1203
1204         reBatteriesCount2 = (await firebaseClient
1205             .Child("EnergyPoints")
1206             .Child(auth.GetUid())
1207             .OnceSingleAsync<EnergyPoints>()).reBatteriesCount;
1208
1209         solarPanelCount2 = (await firebaseClient
1210             .Child("EnergyPoints")
1211             .Child(auth.GetUid())
1212             .OnceSingleAsync<EnergyPoints>()).solarPanelCount;
1213
1214         isolateHomeCount2 = (await firebaseClient
1215             .Child("EnergyPoints")
1216             .Child(auth.GetUid())
1217             .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1218
1219         await firebaseClient
1220             .Child("EnergyPoints")
1221             .Child(auth.GetUid())
1222             .PutAsync(new EnergyPoints()
1223 {
1224     username = username,
1225     points = points2,
1226     numberLogs = numberLogs2,
1227     hangDryCount = hangDry2,
1228     draftSealCount = draftSealCount2,
1229     ductSealCount = ductSealCount2,
1230     efficientThermostatCount = efficientThermostatCount2,
1231     fridgeCount = fridgeCount2,
1232     fullDryerCount = fullDryerCount2,
1233     insulateWaterCount = insulateWaterCount2,
1234     isolateHomeCount = isolateHomeCount2,
1235     ledLightBulbCount = ledLightBulbCount2,
1236     microwaveCount = microwaveCount2,
1237     offSocketCount = offSocketCount2,
1238     reBatteriesCount = reBatteriesCount2,
1239     solarPanelCount = solarPanelCount2,
1240     fullMachineCount = fullMachineCount2
1241 });
1242 }
1243 catch (FirebaseException)
1244 {
1245     username = (await firebaseClient
1246         .Child("users")
1247         .Child(auth.GetUid())
1248         .OnceSingleAsync<Users>()).username;
1249
1250     points2 = AppConstants.fourPoints;
```

```
1251     await firebaseClient
1252         .Child("EnergyPoints")
1253             .Child(auth.GetUid())
1254                 .PutAsync(new EnergyPoints() { username = username, points = points2,
1255     numberOfLogs = 1, microwaveCount = 1 });
1256             }
1257         catch (NullReferenceException)
1258         {
1259             username = (await firebaseClient
1260                 .Child("users")
1261                     .Child(auth.GetUid())
1262                         .OnceSingleAsync<Users>()).username;
1263
1264             points2 = AppConstants.fourPoints;
1265             await firebaseClient
1266                 .Child("EnergyPoints")
1267                     .Child(auth.GetUid())
1268                         .PutAsync(new EnergyPoints() { username = username, points = points2,
1269     numberOfLogs = 1, microwaveCount = 1 });
1270             }
1271         /**
1272          * This function updates the points in the Energy category by four points. It
1273          * also increments the number of logs logged in the Energy
1274          * category by one and increments the number of times this particular action was
1275          * logged by one and sends this data to Firebase.
1276         */
1277         public async void SocketPoints()
1278         {
1279
1280             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1281             quake-default-rtdb.firebaseio.com/");
1282             auth = DependencyService.Get<IAuth>();
1283
1284             try
1285             {
1286                 username = (await firebaseClient
1287                     .Child("users")
1288                         .Child(auth.GetUid())
1289                             .OnceSingleAsync<Users>()).username;
1290
1291                 points2 = (await firebaseClient
1292                     .Child("EnergyPoints")
1293                         .Child(auth.GetUid())
1294                             .OnceSingleAsync<EnergyPoints>()).points;
1295
1296                 points2 = points2 + AppConstants.fourPoints;
1297
1298                 numberOfLogs2 = (await firebaseClient
1299                     .Child("EnergyPoints")
1300                         .Child(auth.GetUid())
1301                             .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
1302
1303                 numberOfLogs2++;
1304
1305                 hangDry2 = (await firebaseClient
1306                     .Child("EnergyPoints")
1307                         .Child(auth.GetUid())
1308                             .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1309
1310                 draftSealCount2 = (await firebaseClient
1311                     .Child("EnergyPoints")
```

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

```

1369         .Child("EnergyPoints")
1370         .Child(auth.GetUid())
1371         .OnceSingleAsync<EnergyPoints>()).isolateHomeCount;
1372
1373     await firebaseClient
1374     .Child("EnergyPoints")
1375     .Child(auth.GetUid())
1376     .PutAsync(new EnergyPoints()
1377     {
1378         username = username,
1379         points = points2,
1380         numberLogs = numberLogs2,
1381         hangDryCount = hangDry2,
1382         draftSealCount = draftSealCount2,
1383         ductSealCount = ductSealCount2,
1384         efficientThermostatCount = efficientThermostatCount2,
1385         fridgeCount = fridgeCount2,
1386         fullDryerCount = fullDryerCount2,
1387         insulateWaterCount = insulateWaterCount2,
1388         isolateHomeCount = isolateHomeCount2,
1389         ledLightBulbCount = ledLightBulbCount2,
1390         microwaveCount = microwaveCount2,
1391         offSocketCount = offSocketCount2,
1392         reBatteriesCount = reBatteriesCount2,
1393         solarPanelCount = solarPanelCount2,
1394         fullMachineCount = fullMachineCount2
1395     });
1396 }
1397 catch (FirebaseException)
1398 {
1399     username = (await firebaseClient
1400     .Child("users")
1401     .Child(auth.GetUid())
1402     .OnceSingleAsync<Users>()).username;
1403
1404     points2 = AppConstants.fourPoints;
1405     await firebaseClient
1406     .Child("EnergyPoints")
1407     .Child(auth.GetUid())
1408     .PutAsync(new EnergyPoints() { username = username, points = points2,
1409     numberLogs = 1, offSocketCount = 1 });
1410 }
1411 catch (NullReferenceException)
1412 {
1413     username = (await firebaseClient
1414     .Child("users")
1415     .Child(auth.GetUid())
1416     .OnceSingleAsync<Users>()).username;
1417
1418     points2 = AppConstants.fourPoints;
1419     await firebaseClient
1420     .Child("EnergyPoints")
1421     .Child(auth.GetUid())
1422     .PutAsync(new EnergyPoints() { username = username, points = points2,
1423     numberLogs = 1, offSocketCount = 1 });
1424 }
1425     /** This function updates the points in the Energy category by six points. It also
1426 increments the number of logs logged in the Energy
1427     * category by one and increments the number of times this particular action was
1428     logged by one and sends this data to Firebase.

```

```
1427  */
1428  public async void ReBatteriesPoints()
1429  {
1430
1431      FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1432      quake-default-rtdb.firebaseio.com/");
1433      auth = DependencyService.Get<IAuth>();
1434
1435      try
1436      {
1437          username = (await firebaseClient
1438              .Child("users")
1439              .Child(auth.GetUserId())
1440              .OnceSingleAsync<Users>()).username;
1441
1442          points2 = (await firebaseClient
1443              .Child("EnergyPoints")
1444              .Child(auth.GetUserId())
1445              .OnceSingleAsync<EnergyPoints>()).points;
1446
1447          points2 = points2 + AppConstants.sixPoints;
1448
1449          numberLogs2 = (await firebaseClient
1450              .Child("EnergyPoints")
1451              .Child(auth.GetUserId())
1452              .OnceSingleAsync<EnergyPoints>()).numberLogs;
1453
1454          numberLogs2++;
1455
1456          hangDry2 = (await firebaseClient
1457              .Child("EnergyPoints")
1458              .Child(auth.GetUserId())
1459              .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1460
1461          draftSealCount2 = (await firebaseClient
1462              .Child("EnergyPoints")
1463              .Child(auth.GetUserId())
1464              .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1465
1466          ductSealCount2 = (await firebaseClient
1467              .Child("EnergyPoints")
1468              .Child(auth.GetUserId())
1469              .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1470
1471          efficientThermostatCount2 = (await firebaseClient
1472              .Child("EnergyPoints")
1473              .Child(auth.GetUserId())
1474              .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
1475
1476          fridgeCount2 = (await firebaseClient
1477              .Child("EnergyPoints")
1478              .Child(auth.GetUserId())
1479              .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1480
1481          fullDryerCount2 = (await firebaseClient
1482              .Child("EnergyPoints")
1483              .Child(auth.GetUserId())
1484              .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1485
1486          fullMachineCount2 = (await firebaseClient
1487              .Child("EnergyPoints")
```

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

```

1548         fullMachineCount = fullMachineCount2
1549     );
1550 }
1551 catch (FirebaseException)
1552 {
1553     username = (await firebaseClient
1554         .Child("users")
1555         .Child(auth.GetUid())
1556         .OnceSingleAsync<Users>()).username;
1557
1558     points2 = AppConstants.fourPoints;
1559     await firebaseClient
1560         .Child("EnergyPoints")
1561         .Child(auth.GetUid())
1562         .PutAsync(new EnergyPoints() { username = username, points = points2,
1563     numberOfLogs = 1, reBatteriesCount = 1 });
1564 }
1565 catch (NullReferenceException)
1566 {
1567     username = (await firebaseClient
1568         .Child("users")
1569         .Child(auth.GetUid())
1570         .OnceSingleAsync<Users>()).username;
1571
1572     points2 = AppConstants.fourPoints;
1573     await firebaseClient
1574         .Child("EnergyPoints")
1575         .Child(auth.GetUid())
1576         .PutAsync(new EnergyPoints() { username = username, points = points2,
1577     numberOfLogs = 1, reBatteriesCount = 1 });
1578 }
1579 /**
1580 * This function updates the points in the Energy category by eight points. It
1581 also increments the number of logs logged in the Energy
1582 * category by one and increments the number of times this particular action was
1583 logged by one and sends this data to Firebase.
1584 */
1585 public async void FridgePoints()
1586 {
1587
1588     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1589     quake-default-rtdb.firebaseio.com/");
1590     auth = DependencyService.Get<IAuth>();
1591
1592     try
1593     {
1594         username = (await firebaseClient
1595             .Child("users")
1596             .Child(auth.GetUid())
1597             .OnceSingleAsync<Users>()).username;
1598
1599         points2 = (await firebaseClient
1600             .Child("EnergyPoints")
1601             .Child(auth.GetUid())
1602             .OnceSingleAsync<EnergyPoints>()).points;
1603
1604         points2 = points2 + AppConstants.eightPoints;
1605
1606         numberOfLogs2 = (await firebaseClient
1607             .Child("EnergyPoints")
1608             .Child(auth.GetUid()))

```

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

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

```
1726             points2 = AppConstants.eightPoints;
1727             await firebaseClient
1728                 .Child("EnergyPoints")
1729                 .Child(auth.GetUid())
1730                 .PutAsync(new EnergyPoints() { username = username, points = points2,
1731     numberOfLogs = 1, fridgeCount = 1 });
1732         }
1733         /**
1734          * This function updates the points in the Energy category by ten points. It also
1735          * increments the number of logs logged in the Energy
1736          * category by one and increments the number of times this particular action was
1737          * logged by one and sends this data to Firebase.
1738         */
1739         public async void SealDraftsPoints()
1740     {
1741
1742         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1743         quake-default-rtdb.firebaseio.com/");
1744         auth = DependencyService.Get<IAuth>();
1745
1746         try
1747     {
1748             username = (await firebaseClient
1749                 .Child("users")
1750                 .Child(auth.GetUid())
1751                 .OnceSingleAsync<Users>()).username;
1752
1753             points2 = (await firebaseClient
1754                 .Child("EnergyPoints")
1755                 .Child(auth.GetUid())
1756                 .OnceSingleAsync<EnergyPoints>()).points;
1757
1758             points2 = points2 + AppConstants.tenPoints;
1759
1760             numberOfLogs2 = (await firebaseClient
1761                 .Child("EnergyPoints")
1762                 .Child(auth.GetUid())
1763                 .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
1764
1765             numberOfLogs2++;
1766
1767             hangDry2 = (await firebaseClient
1768                 .Child("EnergyPoints")
1769                 .Child(auth.GetUid())
1770                 .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1771
1772             draftSealCount2 = (await firebaseClient
1773                 .Child("EnergyPoints")
1774                 .Child(auth.GetUid())
1775                 .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1776
1777             draftSealCount2++;
1778
1779             ductSealCount2 = (await firebaseClient
1780                 .Child("EnergyPoints")
1781                 .Child(auth.GetUid())
1782                 .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1783
1784             efficientThermostatCount2 = (await firebaseClient
1785                 .Child("EnergyPoints")
1786                 .Child(auth.GetUid())
1787                 .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
```

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

```

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

```

```
1902
1903     points2 = (await firebaseClient
1904         .Child("EnergyPoints")
1905         .Child(auth.GetUid())
1906         .OnceSingleAsync<EnergyPoints>()).points;
1907
1908     points2 = points2 + AppConstants.eightPoints;
1909
1910     numberLogs2 = (await firebaseClient
1911         .Child("EnergyPoints")
1912         .Child(auth.GetUid())
1913         .OnceSingleAsync<EnergyPoints>()).numberLogs;
1914
1915     numberLogs2++;
1916
1917     hangDry2 = (await firebaseClient
1918         .Child("EnergyPoints")
1919         .Child(auth.GetUid())
1920         .OnceSingleAsync<EnergyPoints>()).hangDryCount;
1921
1922     draftSealCount2 = (await firebaseClient
1923         .Child("EnergyPoints")
1924         .Child(auth.GetUid())
1925         .OnceSingleAsync<EnergyPoints>()).draftSealCount;
1926
1927     ductSealCount2 = (await firebaseClient
1928         .Child("EnergyPoints")
1929         .Child(auth.GetUid())
1930         .OnceSingleAsync<EnergyPoints>()).ductSealCount;
1931
1932     ductSealCount2++;
1933
1934     efficientThermostatCount2 = (await firebaseClient
1935         .Child("EnergyPoints")
1936         .Child(auth.GetUid())
1937         .OnceSingleAsync<EnergyPoints>()).efficientThermostatCount;
1938
1939     fridgeCount2 = (await firebaseClient
1940         .Child("EnergyPoints")
1941         .Child(auth.GetUid())
1942         .OnceSingleAsync<EnergyPoints>()).fridgeCount;
1943
1944     fullDryerCount2 = (await firebaseClient
1945         .Child("EnergyPoints")
1946         .Child(auth.GetUid())
1947         .OnceSingleAsync<EnergyPoints>()).fullDryerCount;
1948
1949     fullMachineCount2 = (await firebaseClient
1950         .Child("EnergyPoints")
1951         .Child(auth.GetUid())
1952         .OnceSingleAsync<EnergyPoints>()).fullMachineCount;
1953
1954     insulateWaterCount2 = (await firebaseClient
1955         .Child("EnergyPoints")
1956         .Child(auth.GetUid())
1957         .OnceSingleAsync<EnergyPoints>()).insulateWaterCount;
1958
1959     ledLightBulbCount2 = (await firebaseClient
1960         .Child("EnergyPoints")
1961         .Child(auth.GetUid())
1962         .OnceSingleAsync<EnergyPoints>()).ledLightBulbCount;
```

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

```
2024         .PutAsync(new EnergyPoints() { username = username, points = points2,
2025     numberOfLogs = 1, ductSealCount = 1 });
2026     }
2027     catch (NullReferenceException)
2028     {
2029         username = (await firebaseClient
2030             .Child("users")
2031             .Child(auth.GetUid())
2032             .OnceSingleAsync<Users>()).username;
2033
2034         points2 = AppConstants.eightPoints;
2035         await firebaseClient
2036             .Child("EnergyPoints")
2037             .Child(auth.GetUid())
2038             .PutAsync(new EnergyPoints() { username = username, points = points2,
2039     numberOfLogs = 1, ductSealCount = 1 });
2040     }
2041     /** This function updates the points in the Energy category by ten points. It also
2042 increments the number of logs logged in the Energy
2043     * category by one and increments the number of times this particular action was
2044     * logged by one and sends this data to Firebase.
2045     */
2046     public async void SolarPanelPoints()
2047     {
2048
2049         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
2050         quake-default-rtdb.firebaseio.com/");
2051         auth = DependencyService.Get<IAuth>();
2052
2053         try
2054         {
2055             username = (await firebaseClient
2056                 .Child("users")
2057                 .Child(auth.GetUid())
2058                 .OnceSingleAsync<Users>()).username;
2059
2060             points2 = (await firebaseClient
2061                 .Child("EnergyPoints")
2062                 .Child(auth.GetUid())
2063                 .OnceSingleAsync<EnergyPoints>()).points;
2064
2065             points2 = points2 + AppConstants.tenPoints;
2066
2067             numberOfLogs2 = (await firebaseClient
2068                 .Child("EnergyPoints")
2069                 .Child(auth.GetUid())
2070                 .OnceSingleAsync<EnergyPoints>()).numberOfLogs;
2071
2072             numberOfLogs2++;
2073
2074             hangDry2 = (await firebaseClient
2075                 .Child("EnergyPoints")
2076                 .Child(auth.GetUid())
2077                 .OnceSingleAsync<EnergyPoints>()).hangDryCount;
2078
2079             draftSealCount2 = (await firebaseClient
2080                 .Child("EnergyPoints")
2081                 .Child(auth.GetUid())
2082                 .OnceSingleAsync<EnergyPoints>()).draftSealCount;
2083
2084         }
2085     }
2086 }
```

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

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

```
1 /*! \class The FoodAndDrinkPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the FoodAndDrinkPointsUpdate ViewModel Class. It updates the data
8  * for the Food And Drink Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then
10 sending this data to back firebase.
11 *
12 */
13
14 using Application_Green_Quake.Models;
15 using Firebase.Database;
16 using Firebase.Database.Query;
17 using System;
18 using Xamarin.Forms;
19
20 namespace Application_Green_Quake.ViewModels
21 {
22     class FoodAndDrinkPointsUpdate
23     {
24         int points2 = 0;
25         int numberLogs2 = 0;
26         int organicCount2 = 0;
27         int eatAllCount2 = 0;
28         int foodDeliverCount2 = 0;
29         int noMeatCount2 = 0;
30         int ownCoffeeCount2 = 0;
31         int reCoffeeMugCount2 = 0;
32         int saveLeftOversCount2 = 0;
33         int steelStrawCount2 = 0;
34         string username = "";
35
36         IAuth auth;
37         /** This function updates the points in the FoodAndDrink category by eight points.
38          It also increments the number of logs logged in the FoodAndDrink
39          * category by one and increments the number of times this particular action was
40          logged by one and sends this data to Firebase.
41         */
42         public async void OrganicPoints()
43         {
44
45             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
46             quake-default-rtdb.firebaseio.com/");
47             auth = DependencyService.Get<IAuth>();
48
49             try
50             {
51                 username = (await firebaseClient
52                     .Child("users")
53                     .Child(auth.GetUid())
54                     .OnceSingleAsync<Users>()).username;
55
56                 points2 = (await firebaseClient
57                     .Child("FoodAndDrinkPoints")
58                     .Child(auth.GetUid())
59                     .OnceSingleAsync<FoodAndDrinkPoints>()).points;
60
61                 points2 = points2 + AppConstants.eightPoints;
62             }
63         }
64     }
65 }
```

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

```
118         points = points2,
119         numberLogs = numberLogs2,
120         organicCount = organicCount2,
121         eatAllCount = eatAllCount2,
122         foodDeliverCount = foodDeliverCount2,
123         noMeatCount = noMeatCount2,
124         ownCoffeeCount = ownCoffeeCount2,
125         reCoffeeMugCount = reCoffeeMugCount2,
126         saveLeftOversCount = saveLeftOversCount2,
127         steelStrawCount = steelStrawCount2,
128         waterOverFizzyCount = waterOverFizzyCount2,
129     });
130 }
131 catch (FirebaseException)
132 {
133     username = (await firebaseClient
134         .Child("users")
135         .Child(auth.GetUid())
136         .OnceSingleAsync<Users>()).username;
137
138     points2 = AppConstants.eightPoints;
139     await firebaseClient
140         .Child("FoodAndDrinkPoints")
141         .Child(auth.GetUid())
142         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberLogs = 1, organicCount = 1 });
143
144 }
145 catch (NullReferenceException)
146 {
147     username = (await firebaseClient
148         .Child("users")
149         .Child(auth.GetUid())
150         .OnceSingleAsync<Users>()).username;
151
152     points2 = AppConstants.eightPoints;
153     await firebaseClient
154         .Child("FoodAndDrinkPoints")
155         .Child(auth.GetUid())
156         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberLogs = 1, organicCount = 1 });
157
158 }
159 /**
160  * This function updates the points in the FoodAndDrink category by four points.
161  * It also increments the number of logs logged in the FoodAndDrink
162  * category by one and increments the number of times this particular action was
163  * logged by one and sends this data to Firebase.
164  */
165 public async void EatAllPoints()
166 {
167
168     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
169     quake-default-rtdb.firebaseio.com/");
170     auth = DependencyService.Get<IAuth>();
171
172     try
173     {
174         username = (await firebaseClient
175             .Child("users")
176             .Child(auth.GetUid())
177             .OnceSingleAsync<Users>()).username;
```

```
175         points2 = (await firebaseClient
176             .Child("FoodAndDrinkPoints")
177             .Child(auth.GetUid())
178             .OnceSingleAsync<FoodAndDrinkPoints>()).points;
179
180         points2 = points2 + AppConstants.fourPoints;
181
182         numberLogs2 = (await firebaseClient
183             .Child("FoodAndDrinkPoints")
184             .Child(auth.GetUid())
185             .OnceSingleAsync<FoodAndDrinkPoints>()).numberLogs;
186
187         numberLogs2++;
188
189         organicCount2 = (await firebaseClient
190             .Child("FoodAndDrinkPoints")
191             .Child(auth.GetUid())
192             .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
193
194         eatAllCount2 = (await firebaseClient
195             .Child("FoodAndDrinkPoints")
196             .Child(auth.GetUid())
197             .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
198
199         eatAllCount2++;
200
201         foodDeliverCount2 = (await firebaseClient
202             .Child("FoodAndDrinkPoints")
203             .Child(auth.GetUid())
204             .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
205
206         noMeatCount2 = (await firebaseClient
207             .Child("FoodAndDrinkPoints")
208             .Child(auth.GetUid())
209             .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
210
211         ownCoffeeCount2 = (await firebaseClient
212             .Child("FoodAndDrinkPoints")
213             .Child(auth.GetUid())
214             .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
215
216         reCoffeeMugCount2 = (await firebaseClient
217             .Child("FoodAndDrinkPoints")
218             .Child(auth.GetUid())
219             .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
220
221         saveLeftOversCount2 = (await firebaseClient
222             .Child("FoodAndDrinkPoints")
223             .Child(auth.GetUid())
224             .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
225
226         steelStrawCount2 = (await firebaseClient
227             .Child("FoodAndDrinkPoints")
228             .Child(auth.GetUid())
229             .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
230
231         waterOverFizzyCount2 = (await firebaseClient
232             .Child("FoodAndDrinkPoints")
233             .Child(auth.GetUid())
234             .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
235
```

```

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

```

```
293 {
294     username = (await firebaseClient
295         .Child("users")
296         .Child(auth.GetUid())
297         .OnceSingleAsync<Users>()).username;
298
299     points2 = (await firebaseClient
300         .Child("FoodAndDrinkPoints")
301         .Child(auth.GetUid())
302         .OnceSingleAsync<FoodAndDrinkPoints>()).points;
303
304     points2 = points2 + AppConstants.sixPoints;
305
306     numberLogs2 = (await firebaseClient
307         .Child("FoodAndDrinkPoints")
308         .Child(auth.GetUid())
309         .OnceSingleAsync<FoodAndDrinkPoints>()).numberLogs;
310
311     numberLogs2++;
312
313     organicCount2 = (await firebaseClient
314         .Child("FoodAndDrinkPoints")
315         .Child(auth.GetUid())
316         .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
317
318     eatAllCount2 = (await firebaseClient
319         .Child("FoodAndDrinkPoints")
320         .Child(auth.GetUid())
321         .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
322
323     foodDeliverCount2 = (await firebaseClient
324         .Child("FoodAndDrinkPoints")
325         .Child(auth.GetUid())
326         .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
327
328     foodDeliverCount2++;
329
330     noMeatCount2 = (await firebaseClient
331         .Child("FoodAndDrinkPoints")
332         .Child(auth.GetUid())
333         .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
334
335     ownCoffeeCount2 = (await firebaseClient
336         .Child("FoodAndDrinkPoints")
337         .Child(auth.GetUid())
338         .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
339
340     reCoffeeMugCount2 = (await firebaseClient
341         .Child("FoodAndDrinkPoints")
342         .Child(auth.GetUid())
343         .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
344
345     saveLeftOversCount2 = (await firebaseClient
346         .Child("FoodAndDrinkPoints")
347         .Child(auth.GetUid())
348         .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
349
350     steelStrawCount2 = (await firebaseClient
351         .Child("FoodAndDrinkPoints")
352         .Child(auth.GetUid())
353         .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
```

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

```
412
413     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
414     quake-default-rtbd.firebaseio.com/");
415     auth = DependencyService.Get<IAuth>();
416
417     try
418     {
419         username = (await firebaseClient
420             .Child("users")
421             .Child(auth.GetUid())
422             .OnceSingleAsync<Users>()).username;
423
424         points2 = (await firebaseClient
425             .Child("FoodAndDrinkPoints")
426             .Child(auth.GetUid())
427             .OnceSingleAsync<FoodAndDrinkPoints>()).points;
428
429         points2 = points2 + AppConstants.tenPoints;
430
431         numberLogs2 = (await firebaseClient
432             .Child("FoodAndDrinkPoints")
433             .Child(auth.GetUid())
434             .OnceSingleAsync<FoodAndDrinkPoints>()).numberLogs;
435
436         numberLogs2++;
437
438         organicCount2 = (await firebaseClient
439             .Child("FoodAndDrinkPoints")
440             .Child(auth.GetUid())
441             .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
442
443         eatAllCount2 = (await firebaseClient
444             .Child("FoodAndDrinkPoints")
445             .Child(auth.GetUid())
446             .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
447
448         foodDeliverCount2 = (await firebaseClient
449             .Child("FoodAndDrinkPoints")
450             .Child(auth.GetUid())
451             .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
452
453         noMeatCount2 = (await firebaseClient
454             .Child("FoodAndDrinkPoints")
455             .Child(auth.GetUid())
456             .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
457
458         noMeatCount2++;
459
460         ownCoffeeCount2 = (await firebaseClient
461             .Child("FoodAndDrinkPoints")
462             .Child(auth.GetUid())
463             .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
464
465         reCoffeeMugCount2 = (await firebaseClient
466             .Child("FoodAndDrinkPoints")
467             .Child(auth.GetUid())
468             .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
469
470         saveLeftOversCount2 = (await firebaseClient
471             .Child("FoodAndDrinkPoints")
472             .Child(auth.GetUid()))
```

```
472         .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
473
474         steelStrawCount2 = (await firebaseClient
475             .Child("FoodAndDrinkPoints")
476             .Child(auth.GetUid())
477             .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
478
479         waterOverFizzyCount2 = (await firebaseClient
480             .Child("FoodAndDrinkPoints")
481             .Child(auth.GetUid())
482             .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
483
484         await firebaseClient
485             .Child("FoodAndDrinkPoints")
486             .Child(auth.GetUid())
487             .PutAsync(new FoodAndDrinkPoints()
488             {
489                 username = username,
490                 points = points2,
491                 numberofLogs = numberofLogs2,
492                 organicCount = organicCount2,
493                 eatAllCount = eatAllCount2,
494                 foodDeliverCount = foodDeliverCount2,
495                 noMeatCount = noMeatCount2,
496                 ownCoffeeCount = ownCoffeeCount2,
497                 reCoffeeMugCount = reCoffeeMugCount2,
498                 saveLeftOversCount = saveLeftOversCount2,
499                 steelStrawCount = steelStrawCount2,
500                 waterOverFizzyCount = waterOverFizzyCount2,
501             });
502     }
503     catch (FirebaseException)
504     {
505         username = (await firebaseClient
506             .Child("users")
507             .Child(auth.GetUid())
508             .OnceSingleAsync<Users>()).username;
509
510         points2 = AppConstants.tenPoints;
511         await firebaseClient
512             .Child("FoodAndDrinkPoints")
513             .Child(auth.GetUid())
514             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberofLogs = 1, noMeatCount = 1 });
515
516     }
517     catch (NullReferenceException)
518     {
519         username = (await firebaseClient
520             .Child("users")
521             .Child(auth.GetUid())
522             .OnceSingleAsync<Users>()).username;
523
524         points2 = AppConstants.tenPoints;
525         await firebaseClient
526             .Child("FoodAndDrinkPoints")
527             .Child(auth.GetUid())
528             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberofLogs = 1, noMeatCount = 1 });
529     }
530 }
531 /**
 * This function updates the points in the FoodAndDrink category by two points.
```

```
It also increments the number of logs logged in the FoodAndDrink
532     * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
533     */
534     public async void OwnCoffeePoints()
535     {
536
537         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
538         auth = DependencyService.Get<IAuth>();
539
540         try
541         {
542             username = (await firebaseClient
543                 .Child("users")
544                 .Child(auth.GetUid())
545                 .OnceSingleAsync<Users>()).username;
546
547             points2 = (await firebaseClient
548                 .Child("FoodAndDrinkPoints")
549                 .Child(auth.GetUid())
550                 .OnceSingleAsync<FoodAndDrinkPoints>()).points;
551
552             points2 = points2 + AppConstants.twoPoints;
553
554             numberLogs2 = (await firebaseClient
555                 .Child("FoodAndDrinkPoints")
556                 .Child(auth.GetUid())
557                 .OnceSingleAsync<FoodAndDrinkPoints>()).numberLogs;
558
559             numberLogs2++;
560
561             organicCount2 = (await firebaseClient
562                 .Child("FoodAndDrinkPoints")
563                 .Child(auth.GetUid())
564                 .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
565
566             eatAllCount2 = (await firebaseClient
567                 .Child("FoodAndDrinkPoints")
568                 .Child(auth.GetUid())
569                 .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
570
571             foodDeliverCount2 = (await firebaseClient
572                 .Child("FoodAndDrinkPoints")
573                 .Child(auth.GetUid())
574                 .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
575
576             noMeatCount2 = (await firebaseClient
577                 .Child("FoodAndDrinkPoints")
578                 .Child(auth.GetUid())
579                 .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
580
581             ownCoffeeCount2 = (await firebaseClient
582                 .Child("FoodAndDrinkPoints")
583                 .Child(auth.GetUid())
584                 .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
585
586             ownCoffeeCount2++;
587
588             reCoffeeMugCount2 = (await firebaseClient
589                 .Child("FoodAndDrinkPoints")
590                 .Child(auth.GetUid()))
```

```
591     .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
592
593     saveLeftOversCount2 = (await firebaseClient
594         .Child("FoodAndDrinkPoints")
595         .Child(auth.GetUid())
596         .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
597
598     steelStrawCount2 = (await firebaseClient
599         .Child("FoodAndDrinkPoints")
600         .Child(auth.GetUid())
601         .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
602
603     waterOverFizzyCount2 = (await firebaseClient
604         .Child("FoodAndDrinkPoints")
605         .Child(auth.GetUid())
606         .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
607
608     await firebaseClient
609         .Child("FoodAndDrinkPoints")
610         .Child(auth.GetUid())
611         .PutAsync(new FoodAndDrinkPoints()
612     {
613         username = username,
614         points = points2,
615         numberofLogs = numberofLogs2,
616         organicCount = organicCount2,
617         eatAllCount = eatAllCount2,
618         foodDeliverCount = foodDeliverCount2,
619         noMeatCount = noMeatCount2,
620         ownCoffeeCount = ownCoffeeCount2,
621         reCoffeeMugCount = reCoffeeMugCount2,
622         saveLeftOversCount = saveLeftOversCount2,
623         steelStrawCount = steelStrawCount2,
624         waterOverFizzyCount = waterOverFizzyCount2,
625     });
626 }
627 catch (FirebaseException)
628 {
629     username = (await firebaseClient
630         .Child("users")
631         .Child(auth.GetUid())
632         .OnceSingleAsync<Users>()).username;
633
634     points2 = AppConstants.twoPoints;
635     await firebaseClient
636         .Child("FoodAndDrinkPoints")
637         .Child(auth.GetUid())
638         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberofLogs = 1, ownCoffeeCount = 1 });
639
640 }
641 catch (NullReferenceException)
642 {
643     username = (await firebaseClient
644         .Child("users")
645         .Child(auth.GetUid())
646         .OnceSingleAsync<Users>()).username;
647
648     points2 = AppConstants.twoPoints;
649     await firebaseClient
650         .Child("FoodAndDrinkPoints")
```

```
651         .Child(auth.GetUid())
652         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
653             points2, number0fLogs = 1, ownCoffeeCount = 1 });
654     }
655     /** This function updates the points in the FoodAndDrink category by four points.
656     It also increments the number of logs logged in the FoodAndDrink
657     * category by one and increments the number of times this particular action was
658     logged by one and sends this data to Firebase.
659     */
660     public async void ReCoffeeMugPoints()
661     {
662         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
663         quake-default-rtdb.firebaseio.com/");
664         auth = DependencyService.Get<IAuth>();
665
666         try
667         {
668             username = (await firebaseClient
669                 .Child("users")
670                 .Child(auth.GetUid())
671                 .OnceSingleAsync<Users>()).username;
672
673             points2 = (await firebaseClient
674                 .Child("FoodAndDrinkPoints")
675                 .Child(auth.GetUid())
676                 .OnceSingleAsync<FoodAndDrinkPoints>()).points;
677
678             points2 = points2 + AppConstants.fourPoints;
679
680             number0fLogs2 = (await firebaseClient
681                 .Child("FoodAndDrinkPoints")
682                 .Child(auth.GetUid())
683                 .OnceSingleAsync<FoodAndDrinkPoints>()).number0fLogs;
684
685             number0fLogs2++;
686
687             organicCount2 = (await firebaseClient
688                 .Child("FoodAndDrinkPoints")
689                 .Child(auth.GetUid())
690                 .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
691
692             eatAllCount2 = (await firebaseClient
693                 .Child("FoodAndDrinkPoints")
694                 .Child(auth.GetUid())
695                 .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
696
697             foodDeliverCount2 = (await firebaseClient
698                 .Child("FoodAndDrinkPoints")
699                 .Child(auth.GetUid())
700                 .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
701
702             noMeatCount2 = (await firebaseClient
703                 .Child("FoodAndDrinkPoints")
704                 .Child(auth.GetUid())
705                 .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
706
707             ownCoffeeCount2 = (await firebaseClient
708                 .Child("FoodAndDrinkPoints")
709                 .Child(auth.GetUid())
710                 .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
```

```
709
710     reCoffeeMugCount2 = (await firebaseClient
711         .Child("FoodAndDrinkPoints")
712         .Child(auth.GetUid())
713         .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
714
715     reCoffeeMugCount2++;
716
717     saveLeftOversCount2 = (await firebaseClient
718         .Child("FoodAndDrinkPoints")
719         .Child(auth.GetUid())
720         .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
721
722     steelStrawCount2 = (await firebaseClient
723         .Child("FoodAndDrinkPoints")
724         .Child(auth.GetUid())
725         .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
726
727     waterOverFizzyCount2 = (await firebaseClient
728         .Child("FoodAndDrinkPoints")
729         .Child(auth.GetUid())
730         .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
731
732     await firebaseClient
733         .Child("FoodAndDrinkPoints")
734         .Child(auth.GetUid())
735         .PutAsync(new FoodAndDrinkPoints()
736     {
737         username = username,
738         points = points2,
739         numberLogs = numberLogs2,
740         organicCount = organicCount2,
741         eatAllCount = eatAllCount2,
742         foodDeliverCount = foodDeliverCount2,
743         noMeatCount = noMeatCount2,
744         ownCoffeeCount = ownCoffeeCount2,
745         reCoffeeMugCount = reCoffeeMugCount2,
746         saveLeftOversCount = saveLeftOversCount2,
747         steelStrawCount = steelStrawCount2,
748         waterOverFizzyCount = waterOverFizzyCount2,
749     });
750 }
751 catch (FirebaseException)
752 {
753     username = (await firebaseClient
754         .Child("users")
755         .Child(auth.GetUid())
756         .OnceSingleAsync<Users>()).username;
757
758     points2 = AppConstants.fourPoints;
759     await firebaseClient
760         .Child("FoodAndDrinkPoints")
761         .Child(auth.GetUid())
762         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
    points2, numberLogs = 1, reCoffeeMugCount = 1 });
763
764 }
765 catch (NullReferenceException)
766 {
767     username = (await firebaseClient
768         .Child("users")
```

```

769         .Child(auth.GetUid())
770         .OnceSingleAsync<Users>()).username;
771
772         points2 = AppConstants.fourPoints;
773         await firebaseClient
774             .Child("FoodAndDrinkPoints")
775             .Child(auth.GetUid())
776             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
777                 points2, numberofLogs = 1, reCoffeeMugCount = 1 });
778     }
779     /** This function updates the points in the FoodAndDrink category by six points.
It also increments the number of logs logged in the FoodAndDrink
780         * category by one and increments the number of times this particular action was
logged by one and sends this data to Firebase.
781     */
782     public async void SaveLeftOversPoints()
783     {
784
785         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
786         auth = DependencyService.Get<IAuth>();
787
788         try
789         {
790             username = (await firebaseClient
791                 .Child("users")
792                 .Child(auth.GetUid())
793                 .OnceSingleAsync<Users>()).username;
794
795             points2 = (await firebaseClient
796                 .Child("FoodAndDrinkPoints")
797                 .Child(auth.GetUid())
798                 .OnceSingleAsync<FoodAndDrinkPoints>()).points;
799
800             points2 = points2 + AppConstants.sixPoints;
801
802             numberofLogs2 = (await firebaseClient
803                 .Child("FoodAndDrinkPoints")
804                 .Child(auth.GetUid())
805                 .OnceSingleAsync<FoodAndDrinkPoints>()).numberofLogs;
806
807             numberofLogs2++;
808
809             organicCount2 = (await firebaseClient
810                 .Child("FoodAndDrinkPoints")
811                 .Child(auth.GetUid())
812                 .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
813
814             eatAllCount2 = (await firebaseClient
815                 .Child("FoodAndDrinkPoints")
816                 .Child(auth.GetUid())
817                 .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
818
819             foodDeliverCount2 = (await firebaseClient
820                 .Child("FoodAndDrinkPoints")
821                 .Child(auth.GetUid())
822                 .OnceSingleAsync<FoodAndDrinkPoints>()).foodDeliverCount;
823
824             noMeatCount2 = (await firebaseClient
825                 .Child("FoodAndDrinkPoints")
826                 .Child(auth.GetUid()))

```

```
827         .OnceSingleAsync<FoodAndDrinkPoints>()).noMeatCount;
828
829         ownCoffeeCount2 = (await firebaseClient
830             .Child("FoodAndDrinkPoints")
831             .Child(auth.GetUid())
832             .OnceSingleAsync<FoodAndDrinkPoints>()).ownCoffeeCount;
833
834         reCoffeeMugCount2 = (await firebaseClient
835             .Child("FoodAndDrinkPoints")
836             .Child(auth.GetUid())
837             .OnceSingleAsync<FoodAndDrinkPoints>()).reCoffeeMugCount;
838
839         saveLeftOversCount2 = (await firebaseClient
840             .Child("FoodAndDrinkPoints")
841             .Child(auth.GetUid())
842             .OnceSingleAsync<FoodAndDrinkPoints>()).saveLeftOversCount;
843
844         saveLeftOversCount2++;
845
846         steelStrawCount2 = (await firebaseClient
847             .Child("FoodAndDrinkPoints")
848             .Child(auth.GetUid())
849             .OnceSingleAsync<FoodAndDrinkPoints>()).steelStrawCount;
850
851         waterOverFizzyCount2 = (await firebaseClient
852             .Child("FoodAndDrinkPoints")
853             .Child(auth.GetUid())
854             .OnceSingleAsync<FoodAndDrinkPoints>()).waterOverFizzyCount;
855
856         await firebaseClient
857             .Child("FoodAndDrinkPoints")
858             .Child(auth.GetUid())
859             .PutAsync(new FoodAndDrinkPoints()
860             {
861                 username = username,
862                 points = points2,
863                 numberLogs = numberLogs2,
864                 organicCount = organicCount2,
865                 eatAllCount = eatAllCount2,
866                 foodDeliverCount = foodDeliverCount2,
867                 noMeatCount = noMeatCount2,
868                 ownCoffeeCount = ownCoffeeCount2,
869                 reCoffeeMugCount = reCoffeeMugCount2,
870                 saveLeftOversCount = saveLeftOversCount2,
871                 steelStrawCount = steelStrawCount2,
872                 waterOverFizzyCount = waterOverFizzyCount2,
873             });
874     }
875     catch (FirebaseException)
876     {
877         username = (await firebaseClient
878             .Child("users")
879             .Child(auth.GetUid())
880             .OnceSingleAsync<Users>()).username;
881
882         points2 = AppConstants.sixPoints;
883         await firebaseClient
884             .Child("FoodAndDrinkPoints")
885             .Child(auth.GetUid())
886             .PutAsync(new FoodAndDrinkPoints() { username = username, points = points2, numberLogs = 1, saveLeftOversCount = 1 });
887     }
888 }
```

```
887
888     }
889     catch (NullReferenceException)
890     {
891         username = (await firebaseClient
892             .Child("users")
893             .Child(auth.GetUid())
894             .OnceSingleAsync<Users>()).username;
895
896         points2 = AppConstants.sixPoints;
897         await firebaseClient
898             .Child("FoodAndDrinkPoints")
899             .Child(auth.GetUid())
900             .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, numberOfLogs = 1, saveLeftOversCount = 1 });
901     }
902 }
903 /**
904 * This function updates the points in the FoodAndDrink category by four points.
905 * It also increments the number of logs logged in the FoodAndDrink
906 * category by one and increments the number of times this particular action was
907 * logged by one and sends this data to Firebase.
908 */
909 public async void SteelStrawPoints()
910 {
911
912     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
quake-default-rtdb.firebaseio.com/");
913     auth = DependencyService.Get<IAuth>();
914
915     try
916     {
917         username = (await firebaseClient
918             .Child("users")
919             .Child(auth.GetUid())
920             .OnceSingleAsync<Users>()).username;
921
922         points2 = (await firebaseClient
923             .Child("FoodAndDrinkPoints")
924             .Child(auth.GetUid())
925             .OnceSingleAsync<FoodAndDrinkPoints>()).points;
926
927         points2 = points2 + AppConstants.fourPoints;
928
929         numberOfLogs2 = (await firebaseClient
930             .Child("FoodAndDrinkPoints")
931             .Child(auth.GetUid())
932             .OnceSingleAsync<FoodAndDrinkPoints>()).numberOfLogs;
933
934         numberOfLogs2++;
935
936         organicCount2 = (await firebaseClient
937             .Child("FoodAndDrinkPoints")
938             .Child(auth.GetUid())
939             .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
940
941         eatAllCount2 = (await firebaseClient
942             .Child("FoodAndDrinkPoints")
943             .Child(auth.GetUid())
944             .OnceSingleAsync<FoodAndDrinkPoints>()).eatAllCount;
945
946         foodDeliverCount2 = (await firebaseClient
947             .Child("FoodAndDrinkPoints")
```

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

```

1006             points2 = AppConstants.fourPoints;
1007             await firebaseClient
1008                 .Child("FoodAndDrinkPoints")
1009                 .Child(auth.GetUserId())
1010                     .PutAsync(new FoodAndDrinkPoints() { username = username, points =
1011                         points2, numberofLogs = 1, steelStrawCount = 1 });
1012             }
1013         catch (NullReferenceException)
1014         {
1015             username = (await firebaseClient
1016                 .Child("users")
1017                 .Child(auth.GetUserId())
1018                     .OnceSingleAsync<Users>()).username;
1019
1020             points2 = AppConstants.fourPoints;
1021             await firebaseClient
1022                 .Child("FoodAndDrinkPoints")
1023                 .Child(auth.GetUserId())
1024                     .PutAsync(new FoodAndDrinkPoints() { username = username, points =
1025                         points2, numberofLogs = 1, steelStrawCount = 1 });
1026             }
1027             /**
1028             * This function updates the points in the FoodAndDrink category by six points.
1029             * It also increments the number of logs logged in the FoodAndDrink
1030             * category by one and increments the number of times this particular action was
1031             * logged by one and sends this data to Firebase.
1032             */
1033             public async void WaterOverFizzyPoints()
1034             {
1035
1036                 FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1037                 quake-default-rtbd.firebaseio.com/");
1038                 auth = DependencyService.Get<IAuth>();
1039
1040                 try
1041                 {
1042                     username = (await firebaseClient
1043                         .Child("users")
1044                         .Child(auth.GetUserId())
1045                             .OnceSingleAsync<Users>()).username;
1046
1047                     points2 = (await firebaseClient
1048                         .Child("FoodAndDrinkPoints")
1049                         .Child(auth.GetUserId())
1050                             .OnceSingleAsync<FoodAndDrinkPoints>()).points;
1051
1052                     points2 = points2 + AppConstants.sixPoints;
1053
1054                     numberofLogs2 = (await firebaseClient
1055                         .Child("FoodAndDrinkPoints")
1056                         .Child(auth.GetUserId())
1057                             .OnceSingleAsync<FoodAndDrinkPoints>()).numberofLogs;
1058
1059                     numberofLogs2++;
1060
1061                     organicCount2 = (await firebaseClient
1062                         .Child("FoodAndDrinkPoints")
1063                         .Child(auth.GetUserId())
1064                             .OnceSingleAsync<FoodAndDrinkPoints>()).organicCount;
1065
1066                     eatAllCount2 = (await firebaseClient

```

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

```
1124 {
1125     username = (await firebaseClient
1126         .Child("users")
1127         .Child(auth.GetUid())
1128         .OnceSingleAsync<Users>()).username;
1129
1130     points2 = AppConstants.tenPoints;
1131     await firebaseClient
1132         .Child("FoodAndDrinkPoints")
1133         .Child(auth.GetUid())
1134         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, number0fLogs = 1, waterOverFizzyCount = 1 });
1135 }
1136 }
1137 catch (NullReferenceException)
1138 {
1139     username = (await firebaseClient
1140         .Child("users")
1141         .Child(auth.GetUid())
1142         .OnceSingleAsync<Users>()).username;
1143
1144     points2 = AppConstants.tenPoints;
1145     await firebaseClient
1146         .Child("FoodAndDrinkPoints")
1147         .Child(auth.GetUid())
1148         .PutAsync(new FoodAndDrinkPoints() { username = username, points =
points2, number0fLogs = 1, waterOverFizzyCount = 1 });
1149 }
1150 }
1151 }
1152 }
```

```
1 /*! \class The GetAchievementsData ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the GetAchievementsData ViewModel Class. It gets data that is
8  * needed for Achievements from firebase.
9  */
10 using System;
11 using Application_Green_Quake.Models;
12 using Firebase.Database;
13 using Firebase.Database.Query;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class GetAchievementsData
19     {
20         IAuth auth;
21         public static int fixCount = 0;
22         public static int wSShowerHeadCount = 0;
23         public static int waterOverFizzyCount = 0;
24         public static int createGroupCount = 0;
25         public static int communityCount = 0;
26         public static int donateCount = 0;
27         public static int groupCount = 0;
28         public static int shareCount = 0;
29         public static int awarenessCount = 0;
30         public static int fullDryerCount = 0;
31         public static int insulateWaterCount = 0;
32         public static int efficientThermostatCount = 0;
33         public static int isolateHomeCount = 0;
34         public static int ledLightBulbCount = 0;
35         public static int microwaveCount = 0;
36         public static int offSocketCount = 0;
37         public static int reBatteriesCount = 0;
38         public static int reBatCount = 0;
39         public static int fridgeCount = 0;
40         public static int draftSealCount = 0;
41         public static int ductSealCount = 0;
42         public static int solarPanelCount = 0;
43         public static int eatAllCount = 0;
44         public static int foodDeliverCount = 0;
45         public static int noMeatCount = 0;
46         public static int ownCoffeeCount = 0;
47         public static int reCoffeeMugCount = 0;
48         public static int saveLeftOversCount = 0;
49         public static int steelStrawCount = 0;
50         public static int brushingCount = 0;
51         public static int fullWasherCount = 0;
52         public static int showerCount = 0;
53         public static int timedShowerCount = 0;
54         public static int offLigtsCount = 0;
55         public static int matchesCount = 0;
56         public static int airOutCount = 0;
57         public static int nonHarmCount = 0;
58         public static int outsideCount = 0;
59         public static int plantIntoHomeCount = 0;
60         public static int toiletFlushCount = 0;
61         public static int campingCount = 0;
```

```

60  public static int picnicCount = 0;
61  public static int plantBushCount = 0;
62  public static int plantFlowerCount = 0;
63  public static int plantTreeCount = 0;
64  public static int scoopCount = 0;
65  public static int fruitGardenCount = 0;
66  public static int herbGardenCount = 0;
67  public static int vegetableGardenCount = 0;
68  public static int birdFeederCount = 0;
69  public static int clothNapkinCount = 0;
70  public static int clothTowelCount = 0;
71  public static int applianceCount = 0;
72  public static int productCount = 0;
73  public static int toothbrushCount = 0;
74  public static int clothesCount = 0;
75  public static int localCount = 0;
76  public static int looseLeafCount = 0;
77  public static int organicFoodCount = 0;
78  public static int reusableCount = 0;
79  public static int reBagCount = 0;
80  public static int carpoolCount = 0;
81  public static int cycleCount = 0;
82  public static int ecoCarCount = 0;
83  public static int transportCount = 0;
84  public static int walkCount = 0;
85  public static int billsCount = 0;
86  public static int compostCount = 0;
87  public static int setUpRecyclingBinCount = 0;
88  public static int bioBinBagsCount = 0;
89  public static int recyclingBinCount = 0;
90  public static int cisternCount = 0;
91  public static int rainBarrelCount = 0;
92  public static int reWaterCount = 0;
93  public static int showerBucketCount = 0;
94  public static int paperCount = 0;
95  public static int offElectronicsCount = 0;
96  public static int remoteWorkCount = 0;
97  public static int hangDryCount = 0;
98  public static int foodCount = 0;
99
100 /**
101  * This function sets the data that is needed for the Achievements Screen
102 */
103 public async void SetAchievementsData()
104 {
105     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
106     quake-default-rtdb.firebaseio.com/");
107     auth = DependencyService.Get<IAuth>();
108
109     try
110     {
111         fixCount = (await firebaseClient
112             .Child("AdvancedPoints")
113             .Child(auth.GetUid())
114             .OnceSingleAsync<AdvancedPoints>()).fixCount;
115     }
116     catch (Exception e)
117     {
118         Console.WriteLine(e);
119     }
120 }
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
1 /*! \class The GetBadgeData ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the GetBadgeData ViewModel Class. It gets data that is needed for
8  * badges from firebase.
9  */
10 using System;
11 using Application_Green_Quake.Models;
12 using Firebase.Database;
13 using Firebase.Database.Query;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class GetBadgeData
19     {
20         IAuth auth;
21         public static int advancedLog = 0;
22         public static int habitsLog = 0;
23         public static int communityLog = 0;
24         public static int energyLog = 0;
25         public static int foodDrinkLog = 0;
26         public static int homeLog = 0;
27         public static int outdoorsLog = 0;
28         public static int shoppingLog = 0;
29         public static int travelLog = 0;
30         public static int wasteLog = 0;
31         public static int waterLog = 0;
32         public static int workLog = 0;
33
34         /**
35          * This function sets the data that is needed for the badges screen
36         */
37         public async void SetBadgeData()
38         {
39             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
40             quake-default-rtdb.firebaseio.com/");
41             auth = DependencyService.Get<IAuth>();
42
43             try
44             {
45                 advancedLog = (await firebaseClient
46                     .Child("AdvancedPoints")
47                     .Child(auth.GetUid())
48                     .OnceSingleAsync<AdvancedPoints>()).numberOfLogs;
49             }
50             catch (Exception e)
51             {
52                 Console.WriteLine(e);
53             }
54             try
55             {
56                 habitsLog = (await firebaseClient
57                     .Child("HabitsPoints")
58                     .Child(auth.GetUid())
59                     .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
60             }
61             catch (Exception e)
```

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

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

```
1 /*! \class The GetData ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the GetData ViewModel Class. It gets data that is needed for the
8  * applications back end and front end.
9  */
10 using Application_Green_Quake.Models;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using System;
14 using Xamarin.Forms;
15
16 namespace Application_Green_Quake.ViewModels
17 {
18     class GetData
19     {
20         IAuth auth;
21         public static string username = "";
22         public static string bio = "";
23         public static int points = 0;
24         public static int lvl = 0;
25         public static string nation = "";
26
27         /**
28          * This function sets the data to be used for the font end in this application. It
29          * sets the username, bio, points, nation and lvl for the logged in
30          * user. These are saved in global public variables that are used across the
31          * application for front end.
32         */
33         public async void SetData()
34         {
35             try
36             {
37                 FirebaseClient firebaseClient = new FirebaseClient("https://application-
38                 green-quake-default-rtbd.firebaseio.com/");
39                 auth = DependencyService.Get<IAuth>();
40
41                 username = (await firebaseClient
42                     .Child("users")
43                     .Child(auth.GetUid())
44                     .OnceSingleAsync<Users>()).username;
45
46                 bio = (await firebaseClient
47                     .Child("users")
48                     .Child(auth.GetUid())
49                     .OnceSingleAsync<Users>()).bio;
50
51                 nation = (await firebaseClient
52                     .Child("users")
53                     .Child(auth.GetUid())
54                     .OnceSingleAsync<Users>()).nation;
55
56                 points = (await firebaseClient
57                     .Child("Points")
58                     .Child(auth.GetUid())
59                     .OnceSingleAsync<Points>()).points;
60
61                 lvl = (int) Math.Floor((float) points / 10);
62             }
63         }
64     }
65 }
```

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

```
1 /*! \class The HabitsPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the HabitsPointsUpdate ViewModel Class. It updates the data for the
8  * Habits Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 * this data to back firebase.
11 */
12
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class HabitsPointsUpdate
22     {
23         int points2 = 0;
24         int numberLogs2 = 0;
25         int brushingCount2 = 0;
26         int fullWasherCount2 = 0;
27         int showerCount2 = 0;
28         int timedShowerCount2 = 0;
29         int offLightsCount2 = 0;
30         int matchesCount2 = 0;
31
32         string username = "";
33
34         IAuth auth;
35         /** This function updates the points in the Habits category by two points. It also
36          increments the number of logs logged in the Habits
37          * category by one and increments the number of times this particular action was
38          logged by one and sends this data to Firebase.
39         */
40         public async void BrushingPoints()
41         {
42
43             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
44             quake-default-rtdb.firebaseio.com/");
45             auth = DependencyService.Get<IAuth>();
46
47             try
48             {
49                 username = (await firebaseClient
50                     .Child("users")
51                     .Child(auth.GetUid())
52                     .OnceSingleAsync<Users>()).username;
53
54                 points2 = (await firebaseClient
55                     .Child("HabitsPoints")
56                     .Child(auth.GetUid())
57                     .OnceSingleAsync<HabitsPoints>()).points;
58
59                 points2 = points2 + AppConstants.twoPoints;
60
61                 numberLogs2 = (await firebaseClient
62                     .Child("HabitsPoints")
```

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

```
118     await firebaseClient
119         .Child("HabitsPoints")
120         .Child(auth.GetUid())
121         .PutAsync(new HabitsPoints() { username = username, points = points2,
122         numberOfLogs = 1, brushingCount = 1 });
123     }
124     catch (NullReferenceException)
125     {
126         username = (await firebaseClient
127             .Child("users")
128             .Child(auth.GetUid())
129             .OnceSingleAsync<Users>()).username;
130
131         points2 = AppConstants.twoPoints;
132         await firebaseClient
133             .Child("HabitsPoints")
134             .Child(auth.GetUid())
135             .PutAsync(new HabitsPoints() { username = username, points = points2,
136         numberOfLogs = 1, brushingCount = 1 });
137     }
138     /** This function updates the points in the Habits category by eight points. It
139     also increments the number of logs logged in the Habits
140     * category by one and increments the number of times this particular action was
141     logged by one and sends this data to Firebase.
142     */
143     public async void DishWasherFullPoints()
144     {
145
146         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
147         quake-default-rtdb.firebaseio.com/");
148         auth = DependencyService.Get<IAuth>();
149
150         try
151         {
152             username = (await firebaseClient
153                 .Child("users")
154                 .Child(auth.GetUid())
155                 .OnceSingleAsync<Users>()).username;
156
157             points2 = (await firebaseClient
158                 .Child("HabitsPoints")
159                 .Child(auth.GetUid())
160                 .OnceSingleAsync<HabitsPoints>()).points;
161
162             points2 = points2 + AppConstants.eightPoints;
163
164             numberOfLogs2 = (await firebaseClient
165                 .Child("HabitsPoints")
166                 .Child(auth.GetUid())
167                 .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
168
169             numberOfLogs2++;
170
171             brushingCount2 = (await firebaseClient
172                 .Child("HabitsPoints")
173                 .Child(auth.GetUid())
174                 .OnceSingleAsync<HabitsPoints>()).brushingCount;
175
176             fullWasherCount2 = (await firebaseClient
177                 .Child("HabitsPoints")
```

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

```
235     .OnceSingleAsync<Users>()).username;
236 
237     points2 = AppConstants.eightPoints;
238     await firebaseClient
239         .Child("HabitsPoints")
240         .Child(auth.GetUid())
241         .PutAsync(new HabitsPoints() { username = username, points = points2,
242     numberOfLogs = 1, fullWasherCount = 1 });
242     }
243   }
244   /** This function updates the points in the Habits category by six points. It also
245   increments the number of logs logged in the Habits
246   * category by one and increments the number of times this particular action was
247   logged by one and sends this data to Firebase.
248   */
249   public async void ShowerInsteadPoints()
250   {
251 
252     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
253     quake-default-rtdb.firebaseio.com/");
254     auth = DependencyService.Get<IAuth>();
255 
256     try
257     {
258       username = (await firebaseClient
259           .Child("users")
260           .Child(auth.GetUid())
261           .OnceSingleAsync<Users>()).username;
262 
263       points2 = (await firebaseClient
264           .Child("HabitsPoints")
265           .Child(auth.GetUid())
266           .OnceSingleAsync<HabitsPoints>()).points;
267 
268       points2 = points2 + AppConstants.sixPoints;
269 
270       numberOfLogs2 = (await firebaseClient
271           .Child("HabitsPoints")
272           .Child(auth.GetUid())
273           .OnceSingleAsync<HabitsPoints>()).numberOfLogs;
274 
275       numberOfLogs2++;
276 
277       brushingCount2 = (await firebaseClient
278           .Child("HabitsPoints")
279           .Child(auth.GetUid())
280           .OnceSingleAsync<HabitsPoints>()).brushingCount;
281 
282       fullWasherCount2 = (await firebaseClient
283           .Child("HabitsPoints")
284           .Child(auth.GetUid())
285           .OnceSingleAsync<HabitsPoints>()).fullWasherCount;
286 
287       showerCount2 = (await firebaseClient
288           .Child("HabitsPoints")
289           .Child(auth.GetUid())
290           .OnceSingleAsync<HabitsPoints>()).showerCount;
291 
292       showerCount2++;
293 
294       timedShowerCount2 = (await firebaseClient
295           .Child("HabitsPoints")
```

```

293     .Child(auth.GetUid())
294     .OnceSingleAsync<HabitsPoints>()).timedShowerCount;
295 
296     offLigtsCount2 = (await firebaseClient
297     .Child("HabitsPoints")
298     .Child(auth.GetUid())
299     .OnceSingleAsync<HabitsPoints>()).offLigtsCount;
300 
301     matchesCount2 = (await firebaseClient
302     .Child("HabitsPoints")
303     .Child(auth.GetUid())
304     .OnceSingleAsync<HabitsPoints>()).matchesCount;
305 
306     await firebaseClient
307     .Child("HabitsPoints")
308     .Child(auth.GetUid())
309     .PutAsync(new HabitsPoints()
310     {
311         username = username,
312         points = points2,
313         number0fLogs = number0fLogs2,
314         brushingCount = brushingCount2,
315         fullWasherCount = fullWasherCount2,
316         showerCount = showerCount2,
317         timedShowerCount = timedShowerCount2,
318         offLigtsCount = offLigtsCount2,
319         matchesCount = matchesCount2,
320     });
321 }
322 catch (FirebaseException)
323 {
324     username = (await firebaseClient
325     .Child("users")
326     .Child(auth.GetUid())
327     .OnceSingleAsync<Users>()).username;
328 
329     points2 = AppConstants.sixPoints;
330     await firebaseClient
331     .Child("HabitsPoints")
332     .Child(auth.GetUid())
333     .PutAsync(new HabitsPoints() { username = username, points = points2,
number0fLogs = 1, showerCount = 1 });
334 }
335 catch (NullReferenceException)
336 {
337     username = (await firebaseClient
338     .Child("users")
339     .Child(auth.GetUid())
340     .OnceSingleAsync<Users>()).username;
341 
342     points2 = AppConstants.sixPoints;
343     await firebaseClient
344     .Child("HabitsPoints")
345     .Child(auth.GetUid())
346     .PutAsync(new HabitsPoints() { username = username, points = points2,
number0fLogs = 1, showerCount = 1 });
347 }
348 }
349 }
350 /**
This function updates the points in the Habits category by four points. It also
increments the number of logs logged in the Habits
* category by one and increments the number of times this particular action was

```

```
logged by one and sends this data to Firebase.  
352     */  
353     public async void TimedShowerInsteadPoints()  
354     {  
355  
356         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-  
quake-default-rtdb.firebaseio.com/");  
357         auth = DependencyService.Get<IAuth>();  
358  
359         try  
360         {  
361             username = (await firebaseClient  
362                 .Child("users")  
363                 .Child(auth.GetUid())  
364                 .OnceSingleAsync<Users>()).username;  
365  
366             points2 = (await firebaseClient  
367                 .Child("HabitsPoints")  
368                 .Child(auth.GetUid())  
369                 .OnceSingleAsync<HabitsPoints>()).points;  
370  
371             points2 = points2 + AppConstants.fourPoints;  
372  
373             numberofLogs2 = (await firebaseClient  
374                 .Child("HabitsPoints")  
375                 .Child(auth.GetUid())  
376                 .OnceSingleAsync<HabitsPoints>()).numberofLogs;  
377  
378             numberofLogs2++;  
379  
380             brushingCount2 = (await firebaseClient  
381                 .Child("HabitsPoints")  
382                 .Child(auth.GetUid())  
383                 .OnceSingleAsync<HabitsPoints>()).brushingCount;  
384  
385             fullWasherCount2 = (await firebaseClient  
386                 .Child("HabitsPoints")  
387                 .Child(auth.GetUid())  
388                 .OnceSingleAsync<HabitsPoints>()).fullWasherCount;  
389  
390             showerCount2 = (await firebaseClient  
391                 .Child("HabitsPoints")  
392                 .Child(auth.GetUid())  
393                 .OnceSingleAsync<HabitsPoints>()).showerCount;  
394  
395             timedShowerCount2 = (await firebaseClient  
396                 .Child("HabitsPoints")  
397                 .Child(auth.GetUid())  
398                 .OnceSingleAsync<HabitsPoints>()).timedShowerCount;  
399  
400             timedShowerCount2++;  
401  
402             offLightsCount2 = (await firebaseClient  
403                 .Child("HabitsPoints")  
404                 .Child(auth.GetUid())  
405                 .OnceSingleAsync<HabitsPoints>()).offLightsCount;  
406  
407             matchesCount2 = (await firebaseClient  
408                 .Child("HabitsPoints")  
409                 .Child(auth.GetUid())  
410                 .OnceSingleAsync<HabitsPoints>()).matchesCount;
```

```

411
412     await firebaseClient
413         .Child("HabitsPoints")
414             .Child(auth.GetUid())
415                 .PutAsync(new HabitsPoints()
416 {
417     username = username,
418     points = points2,
419     numberLogs = numberLogs2,
420     brushingCount = brushingCount2,
421     fullWasherCount = fullWasherCount2,
422     showerCount = showerCount2,
423     timedShowerCount = timedShowerCount2,
424     offLightsCount = offLightsCount2,
425     matchesCount = matchesCount2,
426 });
427 }
428 catch (FirebaseException)
429 {
430     username = (await firebaseClient
431         .Child("users")
432             .Child(auth.GetUid())
433                 .OnceSingleAsync<Users>()).username;
434
435     points2 = AppConstants.fourPoints;
436     await firebaseClient
437         .Child("HabitsPoints")
438             .Child(auth.GetUid())
439                 .PutAsync(new HabitsPoints() { username = username, points = points2,
440     numberLogs = 1, timedShowerCount = 1 });
441 }
442 catch (NullReferenceException)
443 {
444     username = (await firebaseClient
445         .Child("users")
446             .Child(auth.GetUid())
447                 .OnceSingleAsync<Users>()).username;
448
449     points2 = AppConstants.fourPoints;
450     await firebaseClient
451         .Child("HabitsPoints")
452             .Child(auth.GetUid())
453                 .PutAsync(new HabitsPoints() { username = username, points = points2,
454     numberLogs = 1, timedShowerCount = 1 });
455 }
456 /**
457 * This function updates the points in the Habits category by two points. It also
458 * increments the number of logs logged in the Habits
459 * category by one and increments the number of times this particular action was
460 * logged by one and sends this data to Firebase.
461 */
462 public async void OffLightsPoints()
463 {
464
465     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
466     quake-default-rtdb.firebaseio.com/");
467     auth = DependencyService.Get<IAuth>();
468
469     try
470     {
471         username = (await firebaseClient

```

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

```

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

```

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

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

```
1 /*! \class The HabitsPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the HomePointsUpdate ViewModel Class. It updates the data for the
8  * Home Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 this data to back firebase.
11 *
12 */
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class HomePointsUpdate
22     {
23         int points2 = 0;
24         int numberLogs2 = 0;
25         int airOutCount2 = 0;
26         int nonHarmCount2 = 0;
27         int outsideCount2 = 0;
28         int plantIntoHomeCount2 = 0;
29         int toiletFlushCount2 = 0;
30
31         string username = "";
32
33         IAuth auth;
34         /** This function updates the points in the Home category by two points. It also
35 increments the number of logs logged in the Home
36      * category by one and increments the number of times this particular action was
37      * logged by one and sends this data to Firebase.
38      */
39         public async void AirOutPoints()
40         {
41
42             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
43             quake-default-rtdb.firebaseio.com/");
44             auth = DependencyService.Get<IAuth>();
45
46             try
47             {
48                 username = (await firebaseClient
49                 .Child("users")
50                 .Child(auth.GetUid())
51                 .OnceSingleAsync<Users>()).username;
52
53                 points2 = (await firebaseClient
54                 .Child("HomePoints")
55                 .Child(auth.GetUid())
56                 .OnceSingleAsync<HomePoints>()).points;
57
58                 points2 = points2 + AppConstants.twoPoints;
59
60                 numberLogs2 = (await firebaseClient
61                 .Child("HomePoints")
62                 .Child(auth.GetUid())
63                 .OnceSingleAsync<HomePoints>()).points;
64
65                 numberLogs2 = numberLogs2 + 1;
66
67                 var updateData = new Dictionary<string, object
```

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

```
117     catch (NullReferenceException)
118     {
119         username = (await firebaseClient
120             .Child("users")
121             .Child(auth.GetUid())
122             .OnceSingleAsync<Users>()).username;
123
124         points2 = AppConstants.twoPoints;
125         await firebaseClient
126             .Child("HomePoints")
127             .Child(auth.GetUid())
128             .PutAsync(new HomePoints() { username = username, points = points2,
129             numberOfLogs = 1, airOutCount = 1 });
130     }
131     /** This function updates the points in the Home category by four points. It also
132     increments the number of logs logged in the Home
133     * category by one and increments the number of times this particular action was
134     logged by one and sends this data to Firebase.
135     */
136     public async void NonHarmfulPoints()
137     {
138
139         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
140         quake-default-rtdb.firebaseio.com/");
141         auth = DependencyService.Get<IAuth>();
142
143         try
144         {
145             username = (await firebaseClient
146                 .Child("users")
147                     .Child(auth.GetUid())
148                     .OnceSingleAsync<Users>()).username;
149
150             points2 = (await firebaseClient
151                 .Child("HomePoints")
152                     .Child(auth.GetUid())
153                     .OnceSingleAsync<HomePoints>()).points;
154
155             points2 = points2 + AppConstants.fourPoints;
156
157             numberOfLogs2 = (await firebaseClient
158                 .Child("HomePoints")
159                     .Child(auth.GetUid())
160                     .OnceSingleAsync<HomePoints>()).numberOfLogs;
161
162             numberOfLogs2++;
163
164             airOutCount2 = (await firebaseClient
165                 .Child("HomePoints")
166                     .Child(auth.GetUid())
167                     .OnceSingleAsync<HomePoints>()).airOutCount;
168
169             nonHarmCount2 = (await firebaseClient
170                 .Child("HomePoints")
171                     .Child(auth.GetUid())
172                     .OnceSingleAsync<HomePoints>()).nonHarmCount;
173
174             nonHarmCount2++;
175
176             outsideCount2 = (await firebaseClient
177                 .Child("HomePoints")
```

```

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

```

```
233     */
234     public async void OutsidePoints()
235     {
236
237         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
238         quake-default-rtdb.firebaseio.com/");
239         auth = DependencyService.Get<IAuth>();
240
241         try
242         {
243             username = (await firebaseClient
244                 .Child("users")
245                 .Child(auth.GetUid())
246                 .OnceSingleAsync<Users>()).username;
247
248             points2 = (await firebaseClient
249                 .Child("HomePoints")
250                 .Child(auth.GetUid())
251                 .OnceSingleAsync<HomePoints>()).points;
252
253             points2 = points2 + AppConstants.twoPoints;
254
255             numberofLogs2 = (await firebaseClient
256                 .Child("HomePoints")
257                 .Child(auth.GetUid())
258                 .OnceSingleAsync<HomePoints>()).numberofLogs;
259
260             numberofLogs2++;
261
262             airOutCount2 = (await firebaseClient
263                 .Child("HomePoints")
264                 .Child(auth.GetUid())
265                 .OnceSingleAsync<HomePoints>()).airOutCount;
266
267             nonHarmCount2 = (await firebaseClient
268                 .Child("HomePoints")
269                 .Child(auth.GetUid())
270                 .OnceSingleAsync<HomePoints>()).nonHarmCount;
271
272             outsideCount2 = (await firebaseClient
273                 .Child("HomePoints")
274                 .Child(auth.GetUid())
275                 .OnceSingleAsync<HomePoints>()).outsideCount;
276
277             outsideCount2++;
278
279             plantIntoHomeCount2 = (await firebaseClient
280                 .Child("HomePoints")
281                 .Child(auth.GetUid())
282                 .OnceSingleAsync<HomePoints>()).plantIntoHomeCount;
283
284             toiletFlushCount2 = (await firebaseClient
285                 .Child("HomePoints")
286                 .Child(auth.GetUid())
287                 .OnceSingleAsync<HomePoints>()).toiletFlushCount;
288
289             await firebaseClient
290                 .Child("HomePoints")
291                 .Child(auth.GetUid())
292                 .PutAsync(new HomePoints()
{
```

```
293         username = username,
294         points = points2,
295         numberOfLogs = numberOfLogs2,
296         airOutCount = airOutCount2,
297         nonHarmCount = nonHarmCount2,
298         outsideCount = outsideCount2,
299         plantIntoHomeCount = plantIntoHomeCount2,
300         toiletFlushCount = toiletFlushCount2,
301     });
302 }
303 catch (FirebaseException)
304 {
305     username = (await firebaseClient
306     .Child("users")
307     .Child(auth.GetUid())
308     .OnceSingleAsync<Users>()).username;
309
310     points2 = AppConstants.twoPoints;
311     await firebaseClient
312     .Child("HomePoints")
313     .Child(auth.GetUid())
314     .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, outsideCount = 1 });
315
316 }
317 catch (NullReferenceException)
318 {
319     username = (await firebaseClient
320     .Child("users")
321     .Child(auth.GetUid())
322     .OnceSingleAsync<Users>()).username;
323
324     points2 = AppConstants.twoPoints;
325     await firebaseClient
326     .Child("HomePoints")
327     .Child(auth.GetUid())
328     .PutAsync(new HomePoints() { username = username, points = points2,
numberOfLogs = 1, outsideCount = 1 });
329 }
330 }
331 /**
332  * This function updates the points in the Home category by four points. It also
333  * increments the number of logs logged in the Home
334  * category by one and increments the number of times this particular action was
335  * logged by one and sends this data to Firebase.
336 */
337 public async void PlantsInsidePoints()
338 {
339
340     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
341     quake-default-rtdb.firebaseio.com/");
342     auth = DependencyService.Get<IAuth>();
343
344     try
345     {
346         username = (await firebaseClient
347         .Child("users")
348         .Child(auth.GetUid())
349         .OnceSingleAsync<Users>()).username;
350
351         points2 = (await firebaseClient
352         .Child("HomePoints")
353         .Child(auth.GetUid())
354         .PutAsync(new HomePoints() { username = username, points = points2,
355             airOutCount = 1, nonHarmCount = 1, outsideCount = 1, plantIntoHomeCount = 1,
356             toiletFlushCount = 1 }));
357     }
358 }
```

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

```
411     await firebaseClient
412         .Child("HomePoints")
413         .Child(auth.GetUid())
414         .PutAsync(new HomePoints() { username = username, points = points2,
415     numberOfLogs = 1, plantIntoHomeCount = 1 });
416 }
417 catch (NullReferenceException)
418 {
419     username = (await firebaseClient
420         .Child("users")
421         .Child(auth.GetUid())
422         .OnceSingleAsync<Users>()).username;
423
424     points2 = AppConstants.fourPoints;
425     await firebaseClient
426         .Child("HomePoints")
427         .Child(auth.GetUid())
428         .PutAsync(new HomePoints() { username = username, points = points2,
429     numberOfLogs = 1, plantIntoHomeCount = 1 });
430 }
431 /**
432  * This function updates the points in the Home category by four points. It also
433  * increments the number of logs logged in the Home
434  * category by one and increments the number of times this particular action was
435  * logged by one and sends this data to Firebase.
436 */
437 public async void ToiletPoints()
438 {
439
440     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
441     quake-default-rtdb.firebaseio.com/");
442     auth = DependencyService.Get<IAuth>();
443
444     try
445     {
446         username = (await firebaseClient
447             .Child("users")
448             .Child(auth.GetUid())
449             .OnceSingleAsync<Users>()).username;
450
451         points2 = (await firebaseClient
452             .Child("HomePoints")
453             .Child(auth.GetUid())
454             .OnceSingleAsync<HomePoints>()).points;
455
456         points2 = points2 + AppConstants.fourPoints;
457
458         numberOfLogs2 = (await firebaseClient
459             .Child("HomePoints")
460             .Child(auth.GetUid())
461             .OnceSingleAsync<HomePoints>()).numberOfLogs;
462
463         numberOfLogs2++;
464
465         airOutCount2 = (await firebaseClient
466             .Child("HomePoints")
467             .Child(auth.GetUid())
468             .OnceSingleAsync<HomePoints>()).airOutCount;
469
470         nonHarmCount2 = (await firebaseClient
471             .Child("HomePoints")
```

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

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

```
1 /*! \class The OutdoorsPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the OutdoorsPointsUpdate ViewModel Class. It updates the data for
8  * the Outdoors Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then
10 sending this data to back firebase.
11 *
12 */
13
14 using Application_Green_Quake.Models;
15 using Firebase.Database;
16 using Firebase.Database.Query;
17 using System;
18 using Xamarin.Forms;
19
20 namespace Application_Green_Quake.ViewModels
21 {
22     class OutdoorsPointsUpdate
23     {
24         int points2 = 0;
25         int numberLogs2 = 0;
26         int campingCount2 = 0;
27         int picnicCount2 = 0;
28         int plantBushCount2 = 0;
29         int plantFlowerCount2 = 0;
30         int plantTreeCount2 = 0;
31         int scoopCount2 = 0;
32         int fruitGardenCount2 = 0;
33         int herbGardenCount2 = 0;
34         int vegetableGardenCount2 = 0;
35         int birdFeederCount2 = 0;
36
37         string username = "";
38
39         IAuth auth;
40         /** This function updates the points in the Outdoors category by ten points. It
41         also increments the number of logs logged in the Outdoors
42         * category by one and increments the number of times this particular action was
43         logged by one and sends this data to Firebase.
44         */
45         public async void CampingPoints()
46         {
47             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
48             quake-default-rtdb.firebaseio.com/");
49             auth = DependencyService.Get<IAuth>();
50
51             try
52             {
53                 username = (await firebaseClient
54                     .Child("users")
55                     .Child(auth.GetUserId())
56                     .OnceSingleAsync<Users>()).username;
57
58                 points2 = (await firebaseClient
59                     .Child("OutdoorsPoints")
60                     .Child(auth.GetUserId())
61                     .OnceSingleAsync<OutdoorsPoints>()).points;
62             }
63         }
64     }
65 }
```

```
57     points2 = points2 + AppConstants.sixPoints;
58
59     numberofLogs2 = (await firebaseClient
60         .Child("OutdoorsPoints")
61         .Child(auth.GetUid())
62         .OnceSingleAsync<OutdoorsPoints>()).numberofLogs;
63
64     numberofLogs2++;
65
66     campingCount2 = (await firebaseClient
67         .Child("OutdoorsPoints")
68         .Child(auth.GetUid())
69         .OnceSingleAsync<OutdoorsPoints>()).campingCount;
70
71     campingCount2++;
72
73     picnicCount2 = (await firebaseClient
74         .Child("OutdoorsPoints")
75         .Child(auth.GetUid())
76         .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
77
78     plantBushCount2 = (await firebaseClient
79         .Child("OutdoorsPoints")
80         .Child(auth.GetUid())
81         .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
82
83     plantTreeCount2 = (await firebaseClient
84         .Child("OutdoorsPoints")
85         .Child(auth.GetUid())
86         .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
87
88     plantTreeCount2 = (await firebaseClient
89         .Child("OutdoorsPoints")
90         .Child(auth.GetUid())
91         .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
92
93     scoopCount2 = (await firebaseClient
94         .Child("OutdoorsPoints")
95         .Child(auth.GetUid())
96         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
97
98     fruitGardenCount2 = (await firebaseClient
99         .Child("OutdoorsPoints")
100        .Child(auth.GetUid())
101        .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
102
103    herbGardenCount2 = (await firebaseClient
104        .Child("OutdoorsPoints")
105        .Child(auth.GetUid())
106        .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
107
108    vegetableGardenCount2 = (await firebaseClient
109        .Child("OutdoorsPoints")
110        .Child(auth.GetUid())
111        .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
112
113    birdFeederCount2 = (await firebaseClient
114        .Child("OutdoorsPoints")
115        .Child(auth.GetUid())
116        .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
117
```

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

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

```
236         .Child(auth.GetUid())
237         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
238 
239     vegetableGardenCount2 = (await firebaseClient
240         .Child("OutdoorsPoints")
241         .Child(auth.GetUid())
242         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
243 
244     birdFeederCount2 = (await firebaseClient
245         .Child("OutdoorsPoints")
246         .Child(auth.GetUid())
247         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
248 
249     await firebaseClient
250         .Child("OutdoorsPoints")
251         .Child(auth.GetUid())
252         .PutAsync(new OutdoorsPoints()
253     {
254         username = username,
255         points = points2,
256         numberLogs = numberLogs2,
257         campingCount = campingCount2,
258         picnicCount = picnicCount2,
259         plantBushCount = plantBushCount2,
260         plantFlowerCount = plantBushCount2,
261         plantTreeCount = plantTreeCount2,
262         scoopCount = scoopCount2,
263         fruitGardenCount = fruitGardenCount2,
264         herbGardenCount = herbGardenCount2,
265         vegetableGardenCount = vegetableGardenCount2,
266         birdFeederCount = birdFeederCount2,
267 
268     });
269 }
270 catch (FirebaseException)
271 {
272     username = (await firebaseClient
273         .Child("users")
274         .Child(auth.GetUid())
275         .OnceSingleAsync<Users>()).username;
276 
277     points2 = AppConstants.sixPoints;
278     await firebaseClient
279         .Child("OutdoorsPoints")
280         .Child(auth.GetUid())
281         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
282     numberLogs = 1, picnicCount = 1 }); ;
283 }
284 catch (NullReferenceException)
285 {
286     username = (await firebaseClient
287         .Child("users")
288         .Child(auth.GetUid())
289         .OnceSingleAsync<Users>()).username;
290 
291     points2 = AppConstants.sixPoints;
292     await firebaseClient
293         .Child("OutdoorsPoints")
294         .Child(auth.GetUid())
295         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
```

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

```
354
355     scoopCount2 = (await firebaseClient
356         .Child("OutdoorsPoints")
357         .Child(auth.GetUid())
358         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
359
360     fruitGardenCount2 = (await firebaseClient
361         .Child("OutdoorsPoints")
362         .Child(auth.GetUid())
363         .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
364
365     herbGardenCount2 = (await firebaseClient
366         .Child("OutdoorsPoints")
367         .Child(auth.GetUid())
368         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
369
370     vegetableGardenCount2 = (await firebaseClient
371         .Child("OutdoorsPoints")
372         .Child(auth.GetUid())
373         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
374
375     birdFeederCount2 = (await firebaseClient
376         .Child("OutdoorsPoints")
377         .Child(auth.GetUid())
378         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
379
380     await firebaseClient
381         .Child("OutdoorsPoints")
382         .Child(auth.GetUid())
383         .PutAsync(new OutdoorsPoints()
384     {
385         username = username,
386         points = points2,
387         numberLogs = numberLogs2,
388         campingCount = campingCount2,
389         picnicCount = picnicCount2,
390         plantBushCount = plantBushCount2,
391         plantFlowerCount = plantFlowerCount2,
392         plantTreeCount = plantTreeCount2,
393         scoopCount = scoopCount2,
394         fruitGardenCount = fruitGardenCount2,
395         herbGardenCount = herbGardenCount2,
396         vegetableGardenCount = vegetableGardenCount2,
397         birdFeederCount = birdFeederCount2,
398
399     });
400 }
401 catch (FirebaseException)
402 {
403     username = (await firebaseClient
404         .Child("users")
405         .Child(auth.GetUid())
406         .OnceSingleAsync<Users>()).username;
407
408     points2 = AppConstants.eightPoints;
409     await firebaseClient
410         .Child("OutdoorsPoints")
411         .Child(auth.GetUid())
412         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
413     numberLogs = 1, plantBushCount = 1 });
414 }
```

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

```
472     .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
473 
474     plantFlowerCount2 = (await firebaseClient
475         .Child("OutdoorsPoints")
476         .Child(auth.GetUid())
477         .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
478 
479     plantTreeCount2 = (await firebaseClient
480         .Child("OutdoorsPoints")
481         .Child(auth.GetUid())
482         .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
483 
484     plantTreeCount2++;
485 
486     scoopCount2 = (await firebaseClient
487         .Child("OutdoorsPoints")
488         .Child(auth.GetUid())
489         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
490 
491     fruitGardenCount2 = (await firebaseClient
492         .Child("OutdoorsPoints")
493         .Child(auth.GetUid())
494         .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
495 
496     herbGardenCount2 = (await firebaseClient
497         .Child("OutdoorsPoints")
498         .Child(auth.GetUid())
499         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
500 
501     vegetableGardenCount2 = (await firebaseClient
502         .Child("OutdoorsPoints")
503         .Child(auth.GetUid())
504         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
505 
506     birdFeederCount2 = (await firebaseClient
507         .Child("OutdoorsPoints")
508         .Child(auth.GetUid())
509         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
510 
511     await firebaseClient
512         .Child("OutdoorsPoints")
513         .Child(auth.GetUid())
514         .PutAsync(new OutdoorsPoints()
515     {
516         username = username,
517         points = points2,
518         numberLogs = numberLogs2,
519         campingCount = campingCount2,
520         picnicCount = picnicCount2,
521         plantBushCount = plantBushCount2,
522         plantFlowerCount = plantFlowerCount2,
523         plantTreeCount = plantTreeCount2,
524         scoopCount = scoopCount2,
525         fruitGardenCount = fruitGardenCount2,
526         herbGardenCount = herbGardenCount2,
527         vegetableGardenCount = vegetableGardenCount2,
528         birdFeederCount = birdFeederCount2,
529 
530     });
531 }
532 catch (FirebaseException)
```

```
533 {
534     username = (await firebaseClient
535         .Child("users")
536         .Child(auth.GetUid())
537         .OnceSingleAsync<Users>()).username;
538
539     points2 = AppConstants.tenPoints;
540     await firebaseClient
541         .Child("OutdoorsPoints")
542         .Child(auth.GetUid())
543         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
544     numberOfLogs = 1, plantTreeCount = 1 });
545 }
546 catch (NullReferenceException)
547 {
548     username = (await firebaseClient
549         .Child("users")
550         .Child(auth.GetUid())
551         .OnceSingleAsync<Users>()).username;
552
553     points2 = AppConstants.tenPoints;
554     await firebaseClient
555         .Child("OutdoorsPoints")
556         .Child(auth.GetUid())
557         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
558     numberOfLogs = 1, plantTreeCount = 1 });
559 }
560 /**
561 * This function updates the points in the Outdoors category by eight points. It
562 * also increments the number of logs logged in the Outdoors
563 * category by one and increments the number of times this particular action was
564 * logged by one and sends this data to Firebase.
565 */
566 public async void PlantFlowerPoints()
567 {
568
569     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
570     quake-default-rtdb.firebaseio.com/");
571     auth = DependencyService.Get<IAuth>();
572
573     try
574     {
575         username = (await firebaseClient
576             .Child("users")
577             .Child(auth.GetUid())
578             .OnceSingleAsync<Users>()).username;
579
580         points2 = (await firebaseClient
581             .Child("OutdoorsPoints")
582             .Child(auth.GetUid())
583             .OnceSingleAsync<OutdoorsPoints>()).points;
584
585         points2 = points2 + AppConstants.eightPoints;
586
587         numberOfLogs2 = (await firebaseClient
588             .Child("OutdoorsPoints")
589             .Child(auth.GetUid())
590             .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
591
592         numberOfLogs2++;
593     }
```

```
590     campingCount2 = (await firebaseClient
591         .Child("OutdoorsPoints")
592         .Child(auth.GetUserId())
593         .OnceSingleAsync<OutdoorsPoints>()).campingCount;
594
595     picnicCount2 = (await firebaseClient
596         .Child("OutdoorsPoints")
597         .Child(auth.GetUserId())
598         .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
599
600     plantBushCount2 = (await firebaseClient
601         .Child("OutdoorsPoints")
602         .Child(auth.GetUserId())
603         .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
604
605     plantFlowerCount2 = (await firebaseClient
606         .Child("OutdoorsPoints")
607         .Child(auth.GetUserId())
608         .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
609
610     plantFlowerCount2++;
611
612     plantTreeCount2 = (await firebaseClient
613         .Child("OutdoorsPoints")
614         .Child(auth.GetUserId())
615         .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
616
617     scoopCount2 = (await firebaseClient
618         .Child("OutdoorsPoints")
619         .Child(auth.GetUserId())
620         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
621
622     fruitGardenCount2 = (await firebaseClient
623         .Child("OutdoorsPoints")
624         .Child(auth.GetUserId())
625         .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
626
627     herbGardenCount2 = (await firebaseClient
628         .Child("OutdoorsPoints")
629         .Child(auth.GetUserId())
630         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
631
632     vegetableGardenCount2 = (await firebaseClient
633         .Child("OutdoorsPoints")
634         .Child(auth.GetUserId())
635         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
636
637     birdFeederCount2 = (await firebaseClient
638         .Child("OutdoorsPoints")
639         .Child(auth.GetUserId())
640         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
641
642     await firebaseClient
643         .Child("OutdoorsPoints")
644         .Child(auth.GetUserId())
645         .PutAsync(new OutdoorsPoints()
646     {
647         username = username,
648         points = points2,
649         numberLogs = numberLogs2,
650         campingCount = campingCount2,
```

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

```
708         .Child("OutdoorsPoints")
709         .Child(auth.GetUid())
710         .OnceSingleAsync<OutdoorsPoints>()).points;
711
712         points2 = points2 + AppConstants.fourPoints;
713
714         numberLogs2 = (await firebaseClient
715             .Child("OutdoorsPoints")
716             .Child(auth.GetUid())
717             .OnceSingleAsync<OutdoorsPoints>()).numberLogs;
718
719         numberLogs2++;
720
721         campingCount2 = (await firebaseClient
722             .Child("OutdoorsPoints")
723             .Child(auth.GetUid())
724             .OnceSingleAsync<OutdoorsPoints>()).campingCount;
725
726         picnicCount2 = (await firebaseClient
727             .Child("OutdoorsPoints")
728             .Child(auth.GetUid())
729             .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
730
731         plantBushCount2 = (await firebaseClient
732             .Child("OutdoorsPoints")
733             .Child(auth.GetUid())
734             .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
735
736         plantFlowerCount2 = (await firebaseClient
737             .Child("OutdoorsPoints")
738             .Child(auth.GetUid())
739             .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
740
741         plantTreeCount2 = (await firebaseClient
742             .Child("OutdoorsPoints")
743             .Child(auth.GetUid())
744             .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
745
746         scoopCount2 = (await firebaseClient
747             .Child("OutdoorsPoints")
748             .Child(auth.GetUid())
749             .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
750
751         scoopCount2++;
752
753         fruitGardenCount2 = (await firebaseClient
754             .Child("OutdoorsPoints")
755             .Child(auth.GetUid())
756             .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
757
758         herbGardenCount2 = (await firebaseClient
759             .Child("OutdoorsPoints")
760             .Child(auth.GetUid())
761             .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
762
763         vegetableGardenCount2 = (await firebaseClient
764             .Child("OutdoorsPoints")
765             .Child(auth.GetUid())
766             .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
767
768         birdFeederCount2 = (await firebaseClient
```

```
769         .Child("OutdoorsPoints")
770         .Child(auth.GetUid())
771         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
772 
773     await firebaseClient
774     .Child("OutdoorsPoints")
775     .Child(auth.GetUid())
776     .PutAsync(new OutdoorsPoints()
777     {
778         username = username,
779         points = points2,
780         numberLogs = numberLogs2,
781         campingCount = campingCount2,
782         picnicCount = picnicCount2,
783         plantBushCount = plantBushCount2,
784         plantFlowerCount = plantFlowerCount2,
785         plantTreeCount = plantTreeCount2,
786         scoopCount = scoopCount2,
787         fruitGardenCount = fruitGardenCount2,
788         herbGardenCount = herbGardenCount2,
789         vegetableGardenCount = vegetableGardenCount2,
790         birdFeederCount = birdFeederCount2,
791     });
792 }
793 catch (FirebaseException)
794 {
795     username = (await firebaseClient
796     .Child("users")
797     .Child(auth.GetUid())
798     .OnceSingleAsync<Users>()).username;
799 
800     points2 = AppConstants.fourPoints;
801     await firebaseClient
802     .Child("OutdoorsPoints")
803     .Child(auth.GetUid())
804     .PutAsync(new OutdoorsPoints() { username = username, points = points2,
805     numberLogs = 1, scoopCount = 1 });
806 
807 }
808 catch (NullReferenceException)
809 {
810     username = (await firebaseClient
811     .Child("users")
812     .Child(auth.GetUid())
813     .OnceSingleAsync<Users>()).username;
814 
815     points2 = AppConstants.fourPoints;
816     await firebaseClient
817     .Child("OutdoorsPoints")
818     .Child(auth.GetUid())
819     .PutAsync(new OutdoorsPoints() { username = username, points = points2,
820     numberLogs = 1, scoopCount = 1 });
821 
822     /** This function updates the points in the Outdoors category by ten points. It
823     also increments the number of logs logged in the Outdoors
824     * category by one and increments the number of times this particular action was
825     logged by one and sends this data to Firebase.
826     */
827     public async void FruitGardenPoints()
828     {
```

```
827
828     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
829     quake-default-rtbd.firebaseio.com/");
830     auth = DependencyService.Get<IAuth>();
831
832     try
833     {
834         username = (await firebaseClient
835             .Child("users")
836             .Child(auth.GetUid())
837             .OnceSingleAsync<Users>()).username;
838
839         points2 = (await firebaseClient
840             .Child("OutdoorsPoints")
841             .Child(auth.GetUid())
842             .OnceSingleAsync<OutdoorsPoints>()).points;
843
844         points2 = points2 + AppConstants.tenPoints;
845
846         numberLogs2 = (await firebaseClient
847             .Child("OutdoorsPoints")
848             .Child(auth.GetUid())
849             .OnceSingleAsync<OutdoorsPoints>()).numberLogs;
850
851         numberLogs2++;
852
853         campingCount2 = (await firebaseClient
854             .Child("OutdoorsPoints")
855             .Child(auth.GetUid())
856             .OnceSingleAsync<OutdoorsPoints>()).campingCount;
857
858         picnicCount2 = (await firebaseClient
859             .Child("OutdoorsPoints")
860             .Child(auth.GetUid())
861             .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
862
863         plantBushCount2 = (await firebaseClient
864             .Child("OutdoorsPoints")
865             .Child(auth.GetUid())
866             .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
867
868         plantFlowerCount2 = (await firebaseClient
869             .Child("OutdoorsPoints")
870             .Child(auth.GetUid())
871             .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
872
873         plantTreeCount2 = (await firebaseClient
874             .Child("OutdoorsPoints")
875             .Child(auth.GetUid())
876             .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
877
878         scoopCount2 = (await firebaseClient
879             .Child("OutdoorsPoints")
880             .Child(auth.GetUid())
881             .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
882
883         fruitGardenCount2 = (await firebaseClient
884             .Child("OutdoorsPoints")
885             .Child(auth.GetUid())
886             .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
```

```
887         fruitGardenCount2++;
888
889         herbGardenCount2 = (await firebaseClient
890             .Child("OutdoorsPoints")
891             .Child(auth.GetUid())
892             .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
893
894         vegetableGardenCount2 = (await firebaseClient
895             .Child("OutdoorsPoints")
896             .Child(auth.GetUid())
897             .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
898
899         birdFeederCount2 = (await firebaseClient
900             .Child("OutdoorsPoints")
901             .Child(auth.GetUid())
902             .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
903
904         await firebaseClient
905             .Child("OutdoorsPoints")
906             .Child(auth.GetUid())
907             .PutAsync(new OutdoorsPoints()
908 {
909             username = username,
910             points = points2,
911             numberLogs = numberLogs2,
912             campingCount = campingCount2,
913             picnicCount = picnicCount2,
914             plantBushCount = plantBushCount2,
915             plantFlowerCount = plantFlowerCount2,
916             plantTreeCount = plantTreeCount2,
917             scoopCount = scoopCount2,
918             fruitGardenCount = fruitGardenCount2,
919             herbGardenCount = herbGardenCount2,
920             vegetableGardenCount = vegetableGardenCount2,
921             birdFeederCount = birdFeederCount2,
922
923 });
924 }
925 catch (FirebaseException)
926 {
927     username = (await firebaseClient
928         .Child("users")
929         .Child(auth.GetUid())
930         .OnceSingleAsync<Users>()).username;
931
932     points2 = AppConstants.tenPoints;
933     await firebaseClient
934         .Child("OutdoorsPoints")
935         .Child(auth.GetUid())
936         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
937     numberLogs = 1, fruitGardenCount = 1 }); ;
938
939 }
940 catch (NullReferenceException)
941 {
942     username = (await firebaseClient
943         .Child("users")
944         .Child(auth.GetUid())
945         .OnceSingleAsync<Users>()).username;
946
947     points2 = AppConstants.tenPoints;
```

```
947         await firebaseClient
948             .Child("OutdoorsPoints")
949                 .Child(auth.GetUid())
950                     .PutAsync(new OutdoorsPoints() { username = username, points = points2,
951     numberOfRowsLogs = 1, fruitGardenCount = 1 });
951     }
952   }
953   /** This function updates the points in the Outdoors category by ten points. It
954 also increments the number of logs logged in the Outdoors
955     * category by one and increments the number of times this particular action was
956 logged by one and sends this data to Firebase.
957     */
958   public async void HerbGardenPoints()
959   {
960
961     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
962     quake-default.firebaseio.com/");
963     auth = DependencyService.Get<IAuth>();
964
965     try
966     {
967       username = (await firebaseClient
968           .Child("users")
969               .Child(auth.GetUid())
970                 .OnceSingleAsync<Users>()).username;
971
972       points2 = (await firebaseClient
973           .Child("OutdoorsPoints")
974               .Child(auth.GetUid())
975                 .OnceSingleAsync<OutdoorsPoints>()).points;
976
977       points2 = points2 + AppConstants.tenPoints;
978
979       numberOfRowsLogs2 = (await firebaseClient
980           .Child("OutdoorsPoints")
981               .Child(auth.GetUid())
982                 .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
983
984       numberOfRowsLogs2++;
985
986       campingCount2 = (await firebaseClient
987           .Child("OutdoorsPoints")
988               .Child(auth.GetUid())
989                 .OnceSingleAsync<OutdoorsPoints>()).campingCount;
990
991       picnicCount2 = (await firebaseClient
992           .Child("OutdoorsPoints")
993               .Child(auth.GetUid())
994                 .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
995
996       plantBushCount2 = (await firebaseClient
997           .Child("OutdoorsPoints")
998               .Child(auth.GetUid())
999                 .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
1000
1001       plantFlowerCount2 = (await firebaseClient
1002           .Child("OutdoorsPoints")
1003               .Child(auth.GetUid())
1004                 .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
1005
1006       plantTreeCount2 = (await firebaseClient
1007           .Child("OutdoorsPoints")
```

```
1005     .Child(auth.GetUid())
1006     .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
1007
1008     scoopCount2 = (await firebaseClient
1009         .Child("OutdoorsPoints")
1010         .Child(auth.GetUid())
1011         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
1012
1013     fruitGardenCount2 = (await firebaseClient
1014         .Child("OutdoorsPoints")
1015         .Child(auth.GetUid())
1016         .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
1017
1018     herbGardenCount2 = (await firebaseClient
1019         .Child("OutdoorsPoints")
1020         .Child(auth.GetUid())
1021         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
1022
1023     herbGardenCount2++;
1024
1025     vegetableGardenCount2 = (await firebaseClient
1026         .Child("OutdoorsPoints")
1027         .Child(auth.GetUid())
1028         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
1029
1030     birdFeederCount2 = (await firebaseClient
1031         .Child("OutdoorsPoints")
1032         .Child(auth.GetUid())
1033         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
1034
1035     await firebaseClient
1036         .Child("OutdoorsPoints")
1037         .Child(auth.GetUid())
1038         .PutAsync(new OutdoorsPoints()
1039     {
1040         username = username,
1041         points = points2,
1042         numberLogs = numberLogs2,
1043         campingCount = campingCount2,
1044         picnicCount = picnicCount2,
1045         plantBushCount = plantBushCount2,
1046         plantFlowerCount = plantFlowerCount2,
1047         plantTreeCount = plantTreeCount2,
1048         scoopCount = scoopCount2,
1049         fruitGardenCount = fruitGardenCount2,
1050         herbGardenCount = herbGardenCount2,
1051         vegetableGardenCount = vegetableGardenCount2,
1052         birdFeederCount = birdFeederCount2,
1053
1054     });
1055 }
1056 catch (FirebaseException)
1057 {
1058     username = (await firebaseClient
1059         .Child("users")
1060         .Child(auth.GetUid())
1061         .OnceSingleAsync<Users>()).username;
1062
1063     points2 = AppConstants.tenPoints;
1064     await firebaseClient
1065         .Child("OutdoorsPoints")
```

```
1066             .Child(auth.GetUid())
1067             .PutAsync(new OutdoorsPoints() { username = username, points = points2,
1068     numberOfLogs = 1, herbGardenCount = 1 });
1069         }
1070         catch (NullReferenceException)
1071     {
1072         username = (await firebaseClient
1073             .Child("users")
1074             .Child(auth.GetUid())
1075             .OnceSingleAsync<Users>()).username;
1076
1077         points2 = AppConstants.tenPoints;
1078         await firebaseClient
1079             .Child("OutdoorsPoints")
1080             .Child(auth.GetUid())
1081             .PutAsync(new OutdoorsPoints() { username = username, points = points2,
1082     numberOfLogs = 1, herbGardenCount = 1 });
1083     }
1084     /** This function updates the points in the Outdoors category by ten points. It
1085 also increments the number of logs logged in the Outdoors
1086     * category by one and increments the number of times this particular action was
1087     * logged by one and sends this data to Firebase.
1088     */
1089     public async void VegetableGardenPoints()
1090     {
1091
1092         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1093 quake-default-rtdb.firebaseio.com/");
1094         auth = DependencyService.Get<IAuth>();
1095
1096         try
1097     {
1098             username = (await firebaseClient
1099                 .Child("users")
1100                     .Child(auth.GetUid())
1101                     .OnceSingleAsync<Users>()).username;
1102
1103             points2 = (await firebaseClient
1104                 .Child("OutdoorsPoints")
1105                     .Child(auth.GetUid())
1106                     .OnceSingleAsync<OutdoorsPoints>()).points;
1107
1108             points2 = points2 + AppConstants.tenPoints;
1109
1110             numberOfLogs2 = (await firebaseClient
1111                 .Child("OutdoorsPoints")
1112                     .Child(auth.GetUid())
1113                     .OnceSingleAsync<OutdoorsPoints>()).numberOfLogs;
1114
1115             numberOfLogs2++;
1116
1117             campingCount2 = (await firebaseClient
1118                 .Child("OutdoorsPoints")
1119                     .Child(auth.GetUid())
1120                     .OnceSingleAsync<OutdoorsPoints>()).campingCount;
1121
1122             picnicCount2 = (await firebaseClient
1123                 .Child("OutdoorsPoints")
1124                     .Child(auth.GetUid())
1125                     .OnceSingleAsync<OutdoorsPoints>()).picnicCount;
```

```
1123
1124     plantBushCount2 = (await firebaseClient
1125         .Child("OutdoorsPoints")
1126         .Child(auth.GetUserId())
1127         .OnceSingleAsync<OutdoorsPoints>()).plantBushCount;
1128
1129     plantFlowerCount2 = (await firebaseClient
1130         .Child("OutdoorsPoints")
1131         .Child(auth.GetUserId())
1132         .OnceSingleAsync<OutdoorsPoints>()).plantFlowerCount;
1133
1134     plantTreeCount2 = (await firebaseClient
1135         .Child("OutdoorsPoints")
1136         .Child(auth.GetUserId())
1137         .OnceSingleAsync<OutdoorsPoints>()).plantTreeCount;
1138
1139     scoopCount2 = (await firebaseClient
1140         .Child("OutdoorsPoints")
1141         .Child(auth.GetUserId())
1142         .OnceSingleAsync<OutdoorsPoints>()).scoopCount;
1143
1144     fruitGardenCount2 = (await firebaseClient
1145         .Child("OutdoorsPoints")
1146         .Child(auth.GetUserId())
1147         .OnceSingleAsync<OutdoorsPoints>()).fruitGardenCount;
1148
1149     herbGardenCount2 = (await firebaseClient
1150         .Child("OutdoorsPoints")
1151         .Child(auth.GetUserId())
1152         .OnceSingleAsync<OutdoorsPoints>()).herbGardenCount;
1153
1154     vegetableGardenCount2 = (await firebaseClient
1155         .Child("OutdoorsPoints")
1156         .Child(auth.GetUserId())
1157         .OnceSingleAsync<OutdoorsPoints>()).vegetableGardenCount;
1158
1159     vegetableGardenCount2++;
1160
1161     birdFeederCount2 = (await firebaseClient
1162         .Child("OutdoorsPoints")
1163         .Child(auth.GetUserId())
1164         .OnceSingleAsync<OutdoorsPoints>()).birdFeederCount;
1165
1166     await firebaseClient
1167         .Child("OutdoorsPoints")
1168         .Child(auth.GetUserId())
1169         .PutAsync(new OutdoorsPoints()
1170     {
1171         username = username,
1172         points = points2,
1173         numberofLogs = numberofLogs2,
1174         campingCount = campingCount2,
1175         picnicCount = picnicCount2,
1176         plantBushCount = plantBushCount2,
1177         plantFlowerCount = plantFlowerCount2,
1178         plantTreeCount = plantTreeCount2,
1179         scoopCount = scoopCount2,
1180         fruitGardenCount = fruitGardenCount2,
1181         herbGardenCount = herbGardenCount2,
1182         vegetableGardenCount = vegetableGardenCount2,
1183         birdFeederCount = birdFeederCount2,
```

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

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

```
1302         username = username,
1303         points = points2,
1304         numberLogs = numberLogs2,
1305         campingCount = campingCount2,
1306         picnicCount = picnicCount2,
1307         plantBushCount = plantBushCount2,
1308         plantFlowerCount = plantFlowerCount2,
1309         plantTreeCount = plantTreeCount2,
1310         scoopCount = scoopCount2,
1311         fruitGardenCount = fruitGardenCount2,
1312         herbGardenCount = herbGardenCount2,
1313         vegetableGardenCount = vegetableGardenCount2,
1314         birdFeederCount = birdFeederCount2,
1315
1316     });
1317 }
1318 catch (FirebaseException)
1319 {
1320     username = (await firebaseClient
1321         .Child("users")
1322         .Child(auth.GetUid())
1323         .OnceSingleAsync<Users>()).username;
1324
1325     points2 = AppConstants.tenPoints;
1326     await firebaseClient
1327         .Child("OutdoorsPoints")
1328         .Child(auth.GetUid())
1329         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberLogs = 1, birdFeederCount = 1 });
1330
1331 }
1332 catch (NullReferenceException)
1333 {
1334     username = (await firebaseClient
1335         .Child("users")
1336         .Child(auth.GetUid())
1337         .OnceSingleAsync<Users>()).username;
1338
1339     points2 = AppConstants.tenPoints;
1340     await firebaseClient
1341         .Child("OutdoorsPoints")
1342         .Child(auth.GetUid())
1343         .PutAsync(new OutdoorsPoints() { username = username, points = points2,
numberLogs = 1, birdFeederCount = 1 });
1344
1345 }
1346 }
1347 }
```

```
1 /*! \class The PointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the PointsUpdate ViewModel Class. It updates the data for the
8  * Overall Points of user for the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 * this data to back firebase.
11 *
12 */
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class PointsUpdate
22     {
23         int points2 = 0;
24         string username = "";
25         string theDate = "";
26         long theTime = 0;
27         int theCount = 0;
28         string currentDate = "";
29         long currentTime = 0;
30
31         IAuth auth;
32         /** This function increases the points of a user by ten points. It also stores the
33         current date and time under the Security Checks node and also
34             * increments the security counter if the point update is on the same day and if it
35         is not then it sets it to zero.
36         */
37         public async void UpdateByTenPoints()
38         {
39
40             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
41             quake-default-rtdb.firebaseio.com/");
42             auth = DependencyService.Get<IAuth>();
43
44             currentDate = DateTime.UtcNow.ToString("d");
45             currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
46             try
47             {
48                 theDate = (await firebaseClient
49                     .Child("SecurityChecks")
50                     .Child(auth.GetUserId())
51                     .OnceSingleAsync<SecurityChecks>()).date;
52
53                 theTime = (await firebaseClient
54                     .Child("SecurityChecks")
55                     .Child(auth.GetUserId())
56                     .OnceSingleAsync<SecurityChecks>()).time;
57
58                 theCount = (await firebaseClient
59                     .Child("SecurityChecks")
60                     .Child(auth.GetUserId())
61                     .OnceSingleAsync<SecurityChecks>()).counter;
```

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

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

```
175         .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
176     counter = 1 });
177
178     try
179     {
180         username = (await firebaseClient
181             .Child("users")
182             .Child(auth.GetUid())
183             .OnceSingleAsync<Users>()).username;
184
185         points2 = (await firebaseClient
186             .Child("Points")
187             .Child(auth.GetUid())
188             .OnceSingleAsync<Points>()).points;
189
190         points2 = points2 + AppConstants.eightPoints;
191
192         await firebaseClient
193             .Child("Points")
194             .Child(auth.GetUid())
195             .PutAsync(new Points() { username = username, points = points2 });
196     }
197     catch (FirebaseException)
198     {
199         username = (await firebaseClient
200             .Child("users")
201             .Child(auth.GetUid())
202             .OnceSingleAsync<Users>()).username;
203
204         points2 = AppConstants.eightPoints;
205         await firebaseClient
206             .Child("Points")
207             .Child(auth.GetUid())
208             .PutAsync(new Points() { username = username, points = points2 });
209
210     }
211     catch (NullReferenceException)
212     {
213         username = (await firebaseClient
214             .Child("users")
215             .Child(auth.GetUid())
216             .OnceSingleAsync<Users>()).username;
217
218         points2 = AppConstants.eightPoints;
219         await firebaseClient
220             .Child("Points")
221             .Child(auth.GetUid())
222             .PutAsync(new Points() { username = username, points = points2 });
223     }
224 }
225 /**
226  * This function increases the points of a user by six points. It also stores the
227  * current date and time under the Security Checks node and also
228  * increments the security counter if the point update is on the same day and if it
229  * is not then it sets it to zero.
230  */
231 public async void UpdateBySixPoints()
232 {
233
234     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
235     quake-default-rtdb.firebaseio.com/");
236     auth = DependencyService.Get<IAuth>();
```

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

```
293         .PutAsync(new Points() { username = username, points = points2 });
294     }
295     catch (FirebaseException)
296     {
297         username = (await firebaseClient
298             .Child("users")
299             .Child(auth.GetUid())
300             .OnceSingleAsync<Users>()).username;
301
302         points2 = AppConstants.sixPoints;
303         await firebaseClient
304             .Child("Points")
305             .Child(auth.GetUid())
306             .PutAsync(new Points() { username = username, points = points2 });
307     }
308     catch (NullReferenceException)
309     {
310         username = (await firebaseClient
311             .Child("users")
312             .Child(auth.GetUid())
313             .OnceSingleAsync<Users>()).username;
314
315         points2 = AppConstants.sixPoints;
316         await firebaseClient
317             .Child("Points")
318             .Child(auth.GetUid())
319             .PutAsync(new Points() { username = username, points = points2 });
320     }
321 }
322 */
323 /**
324  * This function increases the points of a user by four points. It also stores the
325  * current date and time under the Security Checks node and also
326  * increments the security counter if the point update is on the same day and if it
327  * is not then it sets it to zero.
328 */
329 public async void UpdateByFourPoints()
330 {
331
332     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
333     quake-default-rtdb.firebaseio.com/");
334     auth = DependencyService.Get<IAuth>();
335
336     currentDate = DateTime.UtcNow.ToString("d");
337     currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
338     try
339     {
340         theDate = (await firebaseClient
341             .Child("SecurityChecks")
342             .Child(auth.GetUid())
343             .OnceSingleAsync<SecurityChecks>()).date;
344
345         theTime = (await firebaseClient
346             .Child("SecurityChecks")
347             .Child(auth.GetUid())
348             .OnceSingleAsync<SecurityChecks>()).time;
349
350         theCount = (await firebaseClient
351             .Child("SecurityChecks")
352             .Child(auth.GetUid())
353             .OnceSingleAsync<SecurityChecks>()).counter;
354
355         if (theDate == currentDate)
```

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

```

411     .Child(auth.GetUid())
412     .OnceSingleAsync<Users>().username;
413
414     points2 = AppConstants.fourPoints;
415     await firebaseClient
416     .Child("Points")
417     .Child(auth.GetUid())
418     .PutAsync(new Points() { username = username, points = points2 });
419   }
420 }
421 /**
422  * This function increases the points of a user by two points. It also stores the
423  * current date and time under the Security Checks node and also
424  * increments the security counter if the point update is on the same day and if it
425  * is not then it sets it to zero.
426 */
427 public async void UpdateByTwoPoints()
428 {
429   FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
430   quake-default-rtdb.firebaseio.com/");
431   auth = DependencyService.Get<IAuth>();
432
433   currentDate = DateTime.UtcNow.ToString("d");
434   currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
435   try
436   {
437     theDate = (await firebaseClient
438     .Child("SecurityChecks")
439     .Child(auth.GetUid())
440     .OnceSingleAsync<SecurityChecks>()).date;
441
442     theTime = (await firebaseClient
443     .Child("SecurityChecks")
444     .Child(auth.GetUid())
445     .OnceSingleAsync<SecurityChecks>()).time;
446
447     theCount = (await firebaseClient
448     .Child("SecurityChecks")
449     .Child(auth.GetUid())
450     .OnceSingleAsync<SecurityChecks>()).counter;
451
452     if (theDate == currentDate)
453     {
454       theCount++;
455     }
456     else
457     {
458       theCount = 0;
459     }
460
461     await firebaseClient
462     .Child("SecurityChecks")
463     .Child(auth.GetUid())
464     .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,
465   counter = theCount });
466
467   }
468   catch (Exception)
469   {
470     await firebaseClient
471     .Child("SecurityChecks")
472     .Child(auth.GetUid())
473     .PutAsync(new SecurityChecks() { date = currentDate, time = currentTime,

```

```
counter = 1 );
}

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

```
1 /*! \class The SecurityMethods ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SecurityMethods ViewModel Class. It performs security checks
8  * for the application whenever a log is made in the application. It
9  * prevents the user from logging more than 15 actions per day and more than 1 action per
10 * 60 seconds.
11 *
12 */
13 using System;
14 using System.Threading.Tasks;
15 using Application_Green_Quake.Models;
16 using Firebase.Database;
17 using Firebase.Database.Query;
18 using Xamarin.Forms;
19
20 namespace Application_Green_Quake.ViewModels
21 {
22     class SecurityMethods
23     {
24         IAuth auth;
25         string theDate = "";
26         long theTime = 0;
27         int theCount = 0;
28         string currentDate = "";
29         long currentTime = 0;
30         private long timeDifference = 0;
31
32         /**
33          * This function gets the date and the count from the SecurityChecks Node in the
34          * database and compares the stored date to the current date. If
35          * the count is 15 and the date is the same as todays date the function returns
36          * true. Otherwise False.
37          * @return value return true/false
38          */
39         public async Task<bool> DayLimitLock()
40         {
41             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
42             quake-default-rtdb.firebaseio.com/");
43             auth = DependencyService.Get<IAuth>();
44
45             currentDate = DateTime.UtcNow.ToString("d");
46             currentTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
47
48             try
49             {
50                 theDate = (await firebaseClient
51                             .Child("SecurityChecks")
52                             .Child(auth.GetUserId())
53                             .OnceSingleAsync<SecurityChecks>()).date;
54
55                 theCount = (await firebaseClient
56                             .Child("SecurityChecks")
57                             .Child(auth.GetUserId())
58                             .OnceSingleAsync<SecurityChecks>()).counter;
59
60                 //If the count in the database is 15 and the time from the date from the
61                 //database is the same as today's date return true.
62                 if (theCount == 15 && theDate == currentDate)
63             }
64         }
65     }
66 }
```

```
56     {
57         return true;
58     }
59
60     return false;
61 }
62 }
63 catch (Exception)
64 {
65     return false;
66 }
67 }
68 /**
69 * This function gets the time from the SecurityChecks Node in the database and
70 compares the stored time to the current time. The time difference
71 * is found by subtracting the time stored in the database from the current time
72 and if the difference is not greater than or equal to 60 seconds then the
73 * function returns true otherwise it returns false.
74 * @return value return true/false
75 */
76 public async Task<bool> TimeLimitLock()
77 {
78     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
79     quake-default-rtdb.firebaseio.com/");
80     auth = DependencyService.Get<IAuth>();
81
82     try
83     {
84         //Get the time stored in the database.
85         theTime = (await firebaseClient
86             .Child("SecurityChecks")
87             .Child(auth.GetUserId())
88             .OnceSingleAsync<SecurityChecks>()).time;
89
90         timeDifference = currentTime - theTime;
91
92         if (timeDifference < 60000)
93         {
94             return true;
95         }
96
97         return false;
98     }
99     catch (Exception)
100    {
101        return false;
102    }
103 }
104 }
105 }
106 }
```

```
1 /*! \class The ShoppingPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ShoppingPointsUpdate ViewModel Class. It updates the data for
8  * the Shopping Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then
10 sending this data to back firebase.
11 *
12 */
13
14 using Application_Green_Quake.Models;
15 using Firebase.Database;
16 using Firebase.Database.Query;
17 using System;
18 using Xamarin.Forms;
19
20 namespace Application_Green_Quake.ViewModels
21 {
22     class ShoppingPointsUpdate
23     {
24         int points2 = 0;
25         int number0fLogs2 = 0;
26         int clothNapkinCount2 = 0;
27         int clothTowelCount2 = 0;
28         int applianceCount2 = 0;
29         int productCount2 = 0;
30         int toothbrushCount2 = 0;
31         int clothesCount2 = 0;
32         int foodCount2 = 0;
33         int localCount2 = 0;
34         int looseLeafCount2 = 0;
35         int organicFoodCount2 = 0;
36         int reusableCount2 = 0;
37         int reBatCount2 = 0;
38         int reBagCount2 = 0;
39
40         string username = "";
41
42         IAuth auth;
43         /** This function updates the points in the Shopping category by two points. It
44         also increments the number of logs logged in the Shopping
45         * category by one and increments the number of times this particular action was
46         logged by one and sends this data to Firebase.
47         */
48         public async void ClothNapkinsPoints()
49         {
50             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
51             quake-default-rtdb.firebaseio.com/");
52             auth = DependencyService.Get<IAuth>();
53
54             try
55             {
56                 username = (await firebaseClient
57                     .Child("users")
58                     .Child(auth.GetUid())
59                     .OnceSingleAsync<Users>()).username;
```

```
57     .Child(auth.GetUid())
58     .OnceSingleAsync<ShoppingPoints>()).points;
59 
60     points2 = points2 + AppConstants.twoPoints;
61 
62     numberLogs2 = (await firebaseClient
63     .Child("ShoppingPoints")
64     .Child(auth.GetUid())
65     .OnceSingleAsync<ShoppingPoints>()).numberLogs;
66 
67     numberLogs2++;
68 
69     clothNapkinCount2 = (await firebaseClient
70     .Child("ShoppingPoints")
71     .Child(auth.GetUid())
72     .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
73 
74     clothNapkinCount2++;
75 
76     clothTowelCount2 = (await firebaseClient
77     .Child("ShoppingPoints")
78     .Child(auth.GetUid())
79     .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
80 
81     applianceCount2 = (await firebaseClient
82     .Child("ShoppingPoints")
83     .Child(auth.GetUid())
84     .OnceSingleAsync<ShoppingPoints>()).applianceCount;
85 
86     productCount2 = (await firebaseClient
87     .Child("ShoppingPoints")
88     .Child(auth.GetUid())
89     .OnceSingleAsync<ShoppingPoints>()).productCount;
90 
91     toothbrushCount2 = (await firebaseClient
92     .Child("ShoppingPoints")
93     .Child(auth.GetUid())
94     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
95 
96     clothesCount2 = (await firebaseClient
97     .Child("ShoppingPoints")
98     .Child(auth.GetUid())
99     .OnceSingleAsync<ShoppingPoints>()).clothesCount;
100 
101    foodCount2 = (await firebaseClient
102     .Child("ShoppingPoints")
103     .Child(auth.GetUid())
104     .OnceSingleAsync<ShoppingPoints>()).foodCount;
105 
106    localCount2 = (await firebaseClient
107     .Child("ShoppingPoints")
108     .Child(auth.GetUid())
109     .OnceSingleAsync<ShoppingPoints>()).localCount;
110 
111    looseLeafCount2 = (await firebaseClient
112     .Child("ShoppingPoints")
113     .Child(auth.GetUid())
114     .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
115 
116    organicFoodCount2 = (await firebaseClient
117     .Child("ShoppingPoints")
```

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

```
178     .OnceSingleAsync<Users>()).username;
179
180     points2 = AppConstants.twoPoints;
181     await firebaseClient
182         .Child("ShoppingPoints")
183         .Child(auth.GetUid())
184         .PutAsync(new ShoppingPoints() { username = username, points = points2,
185     numberOfLogs = 1, clothNapkinCount = 1 });
186 }
187     /** This function updates the points in the Shopping category by two points. It
188 also increments the number of logs logged in the Shopping
189     * category by one and increments the number of times this particular action was
190     * logged by one and sends this data to Firebase.
191     */
192     public async void ClothTowelsPoints()
193     {
194
195         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
196 quake-default-rtdb.firebaseio.com/");
197         auth = DependencyService.Get<IAuth>();
198
199         try
200         {
201             username = (await firebaseClient
202                 .Child("users")
203                 .Child(auth.GetUid())
204                 .OnceSingleAsync<Users>()).username;
205
206             points2 = (await firebaseClient
207                 .Child("ShoppingPoints")
208                 .Child(auth.GetUid())
209                 .OnceSingleAsync<ShoppingPoints>()).points;
210
211             points2 = points2 + AppConstants.twoPoints;
212
213             numberOfLogs2 = (await firebaseClient
214                 .Child("ShoppingPoints")
215                 .Child(auth.GetUid())
216                 .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
217
218             numberOfLogs2++;
219
220             clothNapkinCount2 = (await firebaseClient
221                 .Child("ShoppingPoints")
222                 .Child(auth.GetUid())
223                 .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
224
225             clothTowelCount2 = (await firebaseClient
226                 .Child("ShoppingPoints")
227                 .Child(auth.GetUid())
228                 .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
229
230             clothTowelCount2++;
231
232             applianceCount2 = (await firebaseClient
233                 .Child("ShoppingPoints")
234                 .Child(auth.GetUid())
235                 .OnceSingleAsync<ShoppingPoints>()).applianceCount;
236
237             productCount2 = (await firebaseClient
238                 .Child("ShoppingPoints")
```

```
236         .Child(auth.GetUid())
237         .OnceSingleAsync<ShoppingPoints>()).productCount;
238
239         toothbrushCount2 = (await firebaseClient
240             .Child("ShoppingPoints")
241             .Child(auth.GetUid())
242             .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
243
244         clothesCount2 = (await firebaseClient
245             .Child("ShoppingPoints")
246             .Child(auth.GetUid())
247             .OnceSingleAsync<ShoppingPoints>()).clothesCount;
248
249         foodCount2 = (await firebaseClient
250             .Child("ShoppingPoints")
251             .Child(auth.GetUid())
252             .OnceSingleAsync<ShoppingPoints>()).foodCount;
253
254         localCount2 = (await firebaseClient
255             .Child("ShoppingPoints")
256             .Child(auth.GetUid())
257             .OnceSingleAsync<ShoppingPoints>()).localCount;
258
259         looseLeafCount2 = (await firebaseClient
260             .Child("ShoppingPoints")
261             .Child(auth.GetUid())
262             .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
263
264         organicFoodCount2 = (await firebaseClient
265             .Child("ShoppingPoints")
266             .Child(auth.GetUid())
267             .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
268
269         reusableCount2 = (await firebaseClient
270             .Child("ShoppingPoints")
271             .Child(auth.GetUid())
272             .OnceSingleAsync<ShoppingPoints>()).reusableCount;
273
274         reBatCount2 = (await firebaseClient
275             .Child("ShoppingPoints")
276             .Child(auth.GetUid())
277             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
278
279         reBagCount2 = (await firebaseClient
280             .Child("ShoppingPoints")
281             .Child(auth.GetUid())
282             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
283
284         await firebaseClient
285             .Child("ShoppingPoints")
286             .Child(auth.GetUid())
287             .PutAsync(new ShoppingPoints()
288             {
289                 username = username,
290                 points = points2,
291                 numberofLogs = numberofLogs2,
292                 clothNapkinCount = clothNapkinCount2,
293                 clothTowelCount = clothTowelCount2,
294                 applianceCount = applianceCount2,
295                 productCount = productCount2,
296                 toothbrushCount = toothbrushCount2,
```

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

```
354         .OnceSingleAsync<ShoppingPoints>()).points;
355
356     points2 = points2 + AppConstants.twoPoints;
357
358     numberofLogs2 = (await firebaseClient
359         .Child("ShoppingPoints")
360         .Child(auth.GetUid())
361         .OnceSingleAsync<ShoppingPoints>()).numberofLogs;
362
363     numberofLogs2++;
364
365     clothNapkinCount2 = (await firebaseClient
366         .Child("ShoppingPoints")
367         .Child(auth.GetUid())
368         .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
369
370     clothTowelCount2 = (await firebaseClient
371         .Child("ShoppingPoints")
372         .Child(auth.GetUid())
373         .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
374
375     applianceCount2 = (await firebaseClient
376         .Child("ShoppingPoints")
377         .Child(auth.GetUid())
378         .OnceSingleAsync<ShoppingPoints>()).applianceCount;
379
380     applianceCount2++;
381
382     productCount2 = (await firebaseClient
383         .Child("ShoppingPoints")
384         .Child(auth.GetUid())
385         .OnceSingleAsync<ShoppingPoints>()).productCount;
386
387     toothbrushCount2 = (await firebaseClient
388         .Child("ShoppingPoints")
389         .Child(auth.GetUid())
390         .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
391
392     clothesCount2 = (await firebaseClient
393         .Child("ShoppingPoints")
394         .Child(auth.GetUid())
395         .OnceSingleAsync<ShoppingPoints>()).clothesCount;
396
397     foodCount2 = (await firebaseClient
398         .Child("ShoppingPoints")
399         .Child(auth.GetUid())
400         .OnceSingleAsync<ShoppingPoints>()).foodCount;
401
402     localCount2 = (await firebaseClient
403         .Child("ShoppingPoints")
404         .Child(auth.GetUid())
405         .OnceSingleAsync<ShoppingPoints>()).localCount;
406
407     looseLeafCount2 = (await firebaseClient
408         .Child("ShoppingPoints")
409         .Child(auth.GetUid())
410         .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
411
412     organicFoodCount2 = (await firebaseClient
413         .Child("ShoppingPoints")
414         .Child(auth.GetUid()))
```

```
415     .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
416 
417     reusableCount2 = (await firebaseClient
418         .Child("ShoppingPoints")
419         .Child(auth.GetUid())
420         .OnceSingleAsync<ShoppingPoints>()).reusableCount;
421 
422     reBatCount2 = (await firebaseClient
423         .Child("ShoppingPoints")
424         .Child(auth.GetUid())
425         .OnceSingleAsync<ShoppingPoints>()).reBatCount;
426 
427     reBagCount2 = (await firebaseClient
428         .Child("ShoppingPoints")
429         .Child(auth.GetUid())
430         .OnceSingleAsync<ShoppingPoints>()).reBagCount;
431 
432     await firebaseClient
433         .Child("ShoppingPoints")
434         .Child(auth.GetUid())
435         .PutAsync(new ShoppingPoints()
436     {
437         username = username,
438         points = points2,
439         numberLogs = numberLogs2,
440         clothNapkinCount = clothNapkinCount2,
441         clothTowelCount = clothTowelCount2,
442         applianceCount = applianceCount2,
443         productCount = productCount2,
444         toothbrushCount = toothbrushCount2,
445         clothesCount = clothesCount2,
446         foodCount = foodCount2,
447         localCount = localCount2,
448         looseLeafCount = looseLeafCount2,
449         organicFoodCount = organicFoodCount2,
450         reusableCount = reusableCount2,
451         reBatCount = reBatCount2,
452         reBagCount = reBagCount2,
453     });
454 }
455 catch (FirebaseException)
456 {
457     username = (await firebaseClient
458         .Child("users")
459         .Child(auth.GetUid())
460         .OnceSingleAsync<Users>()).username;
461 
462     points2 = AppConstants.fourPoints;
463     await firebaseClient
464         .Child("ShoppingPoints")
465         .Child(auth.GetUid())
466         .PutAsync(new ShoppingPoints() { username = username, points = points2,
467     numberLogs = 1, applianceCount = 1 });
468 }
469 catch (NullReferenceException)
470 {
471     username = (await firebaseClient
472         .Child("users")
473         .Child(auth.GetUid())
474         .OnceSingleAsync<Users>()).username;
```

```
475
476     points2 = AppConstants.fourPoints;
477     await firebaseClient
478         .Child("ShoppingPoints")
479         .Child(auth.GetUid())
480         .PutAsync(new ShoppingPoints() { username = username, points = points2,
481     numberOfLogs = 1, applianceCount = 1 });
482 }
483     /** This function updates the points in the Shopping category by four points. It
484 also increments the number of logs logged in the Shopping
485     * category by one and increments the number of times this particular action was
486 logged by one and sends this data to Firebase.
487 */
488 public async void ProductPoints()
489 {
490
491     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
492     quake-default-rtbd.firebaseio.com/");
493     auth = DependencyService.Get<IAuth>();
494
495     try
496     {
497         username = (await firebaseClient
498             .Child("users")
499             .Child(auth.GetUid())
500             .OnceSingleAsync<Users>()).username;
501
502         points2 = (await firebaseClient
503             .Child("ShoppingPoints")
504             .Child(auth.GetUid())
505             .OnceSingleAsync<ShoppingPoints>()).points;
506
507         points2 = points2 + AppConstants.fourPoints;
508
509         numberOfLogs2 = (await firebaseClient
510             .Child("ShoppingPoints")
511             .Child(auth.GetUid())
512             .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
513
514         numberOfLogs2++;
515
516         clothNapkinCount2 = (await firebaseClient
517             .Child("ShoppingPoints")
518             .Child(auth.GetUid())
519             .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
520
521         clothTowelCount2 = (await firebaseClient
522             .Child("ShoppingPoints")
523             .Child(auth.GetUid())
524             .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
525
526         applianceCount2 = (await firebaseClient
527             .Child("ShoppingPoints")
528             .Child(auth.GetUid())
529             .OnceSingleAsync<ShoppingPoints>()).applianceCount;
530
531         productCount2 = (await firebaseClient
532             .Child("ShoppingPoints")
533             .Child(auth.GetUid())
534             .OnceSingleAsync<ShoppingPoints>()).productCount;
535
536     }
537 }
```

```
533     productCount2++;
534
535     toothbrushCount2 = (await firebaseClient
536         .Child("ShoppingPoints")
537         .Child(auth.GetUid())
538         .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
539
540     clothesCount2 = (await firebaseClient
541         .Child("ShoppingPoints")
542         .Child(auth.GetUid())
543         .OnceSingleAsync<ShoppingPoints>()).clothesCount;
544
545     foodCount2 = (await firebaseClient
546         .Child("ShoppingPoints")
547         .Child(auth.GetUid())
548         .OnceSingleAsync<ShoppingPoints>()).foodCount;
549
550     localCount2 = (await firebaseClient
551         .Child("ShoppingPoints")
552         .Child(auth.GetUid())
553         .OnceSingleAsync<ShoppingPoints>()).localCount;
554
555     looseLeafCount2 = (await firebaseClient
556         .Child("ShoppingPoints")
557         .Child(auth.GetUid())
558         .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
559
560     organicFoodCount2 = (await firebaseClient
561         .Child("ShoppingPoints")
562         .Child(auth.GetUid())
563         .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
564
565     reusableCount2 = (await firebaseClient
566         .Child("ShoppingPoints")
567         .Child(auth.GetUid())
568         .OnceSingleAsync<ShoppingPoints>()).reusableCount;
569
570     reBatCount2 = (await firebaseClient
571         .Child("ShoppingPoints")
572         .Child(auth.GetUid())
573         .OnceSingleAsync<ShoppingPoints>()).reBatCount;
574
575     reBagCount2 = (await firebaseClient
576         .Child("ShoppingPoints")
577         .Child(auth.GetUid())
578         .OnceSingleAsync<ShoppingPoints>()).reBagCount;
579
580     await firebaseClient
581         .Child("ShoppingPoints")
582         .Child(auth.GetUid())
583         .PutAsync(new ShoppingPoints()
584     {
585         username = username,
586         points = points2,
587         numberofLogs = numberofLogs2,
588         clothNapkinCount = clothNapkinCount2,
589         clothTowelCount = clothTowelCount2,
590         applianceCount = applianceCount2,
591         productCount = productCount2,
592         toothbrushCount = toothbrushCount2,
593         clothesCount = clothesCount2,
```

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

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

```
712
713     reusableCount2 = (await firebaseClient
714         .Child("ShoppingPoints")
715         .Child(auth.GetUid())
716         .OnceSingleAsync<ShoppingPoints>()).reusableCount;
717
718     reBatCount2 = (await firebaseClient
719         .Child("ShoppingPoints")
720         .Child(auth.GetUid())
721         .OnceSingleAsync<ShoppingPoints>()).reBatCount;
722
723     reBagCount2 = (await firebaseClient
724         .Child("ShoppingPoints")
725         .Child(auth.GetUid())
726         .OnceSingleAsync<ShoppingPoints>()).reBagCount;
727
728     await firebaseClient
729         .Child("ShoppingPoints")
730         .Child(auth.GetUid())
731         .PutAsync(new ShoppingPoints()
732     {
733         username = username,
734         points = points2,
735         numberLogs = numberLogs2,
736         clothNapkinCount = clothNapkinCount2,
737         clothTowelCount = clothTowelCount2,
738         applianceCount = applianceCount2,
739         productCount = productCount2,
740         toothbrushCount = toothbrushCount2,
741         clothesCount = clothesCount2,
742         foodCount = foodCount2,
743         localCount = localCount2,
744         looseLeafCount = looseLeafCount2,
745         organicFoodCount = organicFoodCount2,
746         reusableCount = reusableCount2,
747         reBatCount = reBatCount2,
748         reBagCount = reBagCount2,
749     });
750 }
751 catch (FirebaseException)
752 {
753     username = (await firebaseClient
754         .Child("users")
755         .Child(auth.GetUid())
756         .OnceSingleAsync<Users>()).username;
757
758     points2 = AppConstants.sixPoints;
759     await firebaseClient
760         .Child("ShoppingPoints")
761         .Child(auth.GetUid())
762         .PutAsync(new ShoppingPoints() { username = username, points = points2,
763     numberLogs = 1, toothbrushCount = 1 });
764 }
765 catch (NullReferenceException)
766 {
767     username = (await firebaseClient
768         .Child("users")
769         .Child(auth.GetUid())
770         .OnceSingleAsync<Users>()).username;
771 }
```

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

```
830         .Child("ShoppingPoints")
831         .Child(auth.GetUid())
832         .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
833
834         clothesCount2 = (await firebaseClient
835             .Child("ShoppingPoints")
836             .Child(auth.GetUid())
837             .OnceSingleAsync<ShoppingPoints>()).clothesCount;
838
839         clothesCount2++;
840
841         foodCount2 = (await firebaseClient
842             .Child("ShoppingPoints")
843             .Child(auth.GetUid())
844             .OnceSingleAsync<ShoppingPoints>()).foodCount;
845
846         localCount2 = (await firebaseClient
847             .Child("ShoppingPoints")
848             .Child(auth.GetUid())
849             .OnceSingleAsync<ShoppingPoints>()).localCount;
850
851         looseLeafCount2 = (await firebaseClient
852             .Child("ShoppingPoints")
853             .Child(auth.GetUid())
854             .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
855
856         organicFoodCount2 = (await firebaseClient
857             .Child("ShoppingPoints")
858             .Child(auth.GetUid())
859             .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
860
861         reusableCount2 = (await firebaseClient
862             .Child("ShoppingPoints")
863             .Child(auth.GetUid())
864             .OnceSingleAsync<ShoppingPoints>()).reusableCount;
865
866         reBatCount2 = (await firebaseClient
867             .Child("ShoppingPoints")
868             .Child(auth.GetUid())
869             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
870
871         reBagCount2 = (await firebaseClient
872             .Child("ShoppingPoints")
873             .Child(auth.GetUid())
874             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
875
876         await firebaseClient
877             .Child("ShoppingPoints")
878             .Child(auth.GetUid())
879             .PutAsync(new ShoppingPoints()
880 {
881             username = username,
882             points = points2,
883             numberofLogs = numberofLogs2,
884             clothNapkinCount = clothNapkinCount2,
885             clothTowelCount = clothTowelCount2,
886             applianceCount = applianceCount2,
887             productCount = productCount2,
888             toothbrushCount = toothbrushCount2,
889             clothesCount = clothesCount2,
890             foodCount = foodCount2,
```

```

91         localCount = localCount2,
92         looseLeafCount = looseLeafCount2,
93         organicFoodCount = organicFoodCount2,
94         reusableCount = reusableCount2,
95         reBatCount = reBatCount2,
96         reBagCount = reBagCount2,
97     });
98 }
99 catch (FirebaseException)
100 {
101     username = (await firebaseClient
102         .Child("users")
103         .Child(auth.GetUid())
104         .OnceSingleAsync<Users>()).username;
105
106     points2 = AppConstants.tenPoints;
107     await firebaseClient
108         .Child("ShoppingPoints")
109         .Child(auth.GetUid())
110         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, clothesCount = 1 });
111 }
112 catch (NullReferenceException)
113 {
114     username = (await firebaseClient
115         .Child("users")
116         .Child(auth.GetUid())
117         .OnceSingleAsync<Users>()).username;
118
119     points2 = AppConstants.tenPoints;
120     await firebaseClient
121         .Child("ShoppingPoints")
122         .Child(auth.GetUid())
123         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, clothesCount = 1 });
124 }
125 }
126 /**
127  * This function updates the points in the Shopping category by six points. It
128  * also increments the number of logs logged in the Shopping
129  * category by one and increments the number of times this particular action was
130  * logged by one and sends this data to Firebase.
131 */
132 public async void FoodInBulkPoints()
133 {
134
135     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
136     quake-default-rtdb.firebaseio.com/");
137     auth = DependencyService.Get<IAuth>();
138
139     try
140     {
141         username = (await firebaseClient
142             .Child("users")
143             .Child(auth.GetUid())
144             .OnceSingleAsync<Users>()).username;
145
146         points2 = (await firebaseClient
147             .Child("ShoppingPoints")
148             .Child(auth.GetUid())
149             .OnceSingleAsync<ShoppingPoints>()).points;
150     }
151 }

```

```
948     points2 = points2 + AppConstants.sixPoints;
949
950     numberofLogs2 = (await firebaseClient
951         .Child("ShoppingPoints")
952         .Child(auth.GetUid())
953         .OnceSingleAsync<ShoppingPoints>()).numberofLogs;
954
955     numberofLogs2++;
956
957     clothNapkinCount2 = (await firebaseClient
958         .Child("ShoppingPoints")
959         .Child(auth.GetUid())
960         .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
961
962     clothTowelCount2 = (await firebaseClient
963         .Child("ShoppingPoints")
964         .Child(auth.GetUid())
965         .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
966
967     applianceCount2 = (await firebaseClient
968         .Child("ShoppingPoints")
969         .Child(auth.GetUid())
970         .OnceSingleAsync<ShoppingPoints>()).applianceCount;
971
972     productCount2 = (await firebaseClient
973         .Child("ShoppingPoints")
974         .Child(auth.GetUid())
975         .OnceSingleAsync<ShoppingPoints>()).productCount;
976
977     toothbrushCount2 = (await firebaseClient
978         .Child("ShoppingPoints")
979         .Child(auth.GetUid())
980         .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
981
982     clothesCount2 = (await firebaseClient
983         .Child("ShoppingPoints")
984         .Child(auth.GetUid())
985         .OnceSingleAsync<ShoppingPoints>()).clothesCount;
986
987     foodCount2 = (await firebaseClient
988         .Child("ShoppingPoints")
989         .Child(auth.GetUid())
990         .OnceSingleAsync<ShoppingPoints>()).foodCount;
991
992     foodCount2++;
993
994     localCount2 = (await firebaseClient
995         .Child("ShoppingPoints")
996         .Child(auth.GetUid())
997         .OnceSingleAsync<ShoppingPoints>()).localCount;
998
999     looseLeafCount2 = (await firebaseClient
1000         .Child("ShoppingPoints")
1001         .Child(auth.GetUid())
1002         .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1003
1004     organicFoodCount2 = (await firebaseClient
1005         .Child("ShoppingPoints")
1006         .Child(auth.GetUid())
1007         .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1008
```

```
1009         reusableCount2 = (await firebaseClient
1010             .Child("ShoppingPoints")
1011             .Child(auth.GetUid())
1012             .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1013
1014         reBatCount2 = (await firebaseClient
1015             .Child("ShoppingPoints")
1016             .Child(auth.GetUid())
1017             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1018
1019         reBagCount2 = (await firebaseClient
1020             .Child("ShoppingPoints")
1021             .Child(auth.GetUid())
1022             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1023
1024         await firebaseClient
1025             .Child("ShoppingPoints")
1026             .Child(auth.GetUid())
1027             .PutAsync(new ShoppingPoints()
1028 {
1029     username = username,
1030     points = points2,
1031     numberOfLogs = numberOfLogs2,
1032     clothNapkinCount = clothNapkinCount2,
1033     clothTowelCount = clothTowelCount2,
1034     applianceCount = applianceCount2,
1035     productCount = productCount2,
1036     toothbrushCount = toothbrushCount2,
1037     clothesCount = clothesCount2,
1038     foodCount = foodCount2,
1039     localCount = localCount2,
1040     looseLeafCount = looseLeafCount2,
1041     organicFoodCount = organicFoodCount2,
1042     reusableCount = reusableCount2,
1043     reBatCount = reBatCount2,
1044     reBagCount = reBagCount2,
1045 });
1046 }
1047 catch (FirebaseException)
1048 {
1049     username = (await firebaseClient
1050         .Child("users")
1051         .Child(auth.GetUid())
1052         .OnceSingleAsync<Users>()).username;
1053
1054     points2 = AppConstants.sixPoints;
1055     await firebaseClient
1056         .Child("ShoppingPoints")
1057         .Child(auth.GetUid())
1058         .PutAsync(new ShoppingPoints() { username = username, points = points2,
1059     numberOfLogs = 1, foodCount = 1 });
1060
1061     }
1062     catch (NullReferenceException)
1063     {
1064         username = (await firebaseClient
1065             .Child("users")
1066             .Child(auth.GetUid())
1067             .OnceSingleAsync<Users>()).username;
1068
1069         points2 = AppConstants.sixPoints;
```

```
1069         await firebaseClient
1070             .Child("ShoppingPoints")
1071                 .Child(auth.GetUid())
1072                     .PutAsync(new ShoppingPoints() { username = username, points = points2,
1073     numberOfLogs = 1, foodCount = 1 });
1074     }
1075     /** This function updates the points in the Shopping category by eight points. It
1076 also increments the number of logs logged in the Shopping
1077     * category by one and increments the number of times this particular action was
1078     * logged by one and sends this data to Firebase.
1079     */
1080     public async void LocalProductPoints()
1081     {
1082
1083         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1084     quake-default.firebaseio.com/");
1085         auth = DependencyService.Get<IAuth>();
1086
1087         try
1088     {
1089         username = (await firebaseClient
1090             .Child("users")
1091                 .Child(auth.GetUid())
1092                     .OnceSingleAsync<Users>()).username;
1093
1094         points2 = (await firebaseClient
1095             .Child("ShoppingPoints")
1096                 .Child(auth.GetUid())
1097                     .OnceSingleAsync<ShoppingPoints>()).points;
1098
1099         points2 = points2 + AppConstants.eightPoints;
1100
1101         numberOfLogs2 = (await firebaseClient
1102             .Child("ShoppingPoints")
1103                 .Child(auth.GetUid())
1104                     .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
1105
1106         numberOfLogs2++;
1107
1108         clothNapkinCount2 = (await firebaseClient
1109             .Child("ShoppingPoints")
1110                 .Child(auth.GetUid())
1111                     .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1112
1113         clothTowelCount2 = (await firebaseClient
1114             .Child("ShoppingPoints")
1115                 .Child(auth.GetUid())
1116                     .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1117
1118         applianceCount2 = (await firebaseClient
1119             .Child("ShoppingPoints")
1120                 .Child(auth.GetUid())
1121                     .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1122
1123         productCount2 = (await firebaseClient
1124             .Child("ShoppingPoints")
1125                 .Child(auth.GetUid())
1126                     .OnceSingleAsync<ShoppingPoints>()).productCount;
1127
1128         toothbrushCount2 = (await firebaseClient
1129             .Child("ShoppingPoints")
```

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

```
1188         looseLeafCount = looseLeafCount2,
1189         organicFoodCount = organicFoodCount2,
1190         reusableCount = reusableCount2,
1191         reBatCount = reBatCount2,
1192         reBagCount = reBagCount2,
1193     });
1194 }
1195 catch (FirebaseException)
1196 {
1197     username = (await firebaseClient
1198         .Child("users")
1199         .Child(auth.GetUid())
1200         .OnceSingleAsync<Users>()).username;
1201
1202     points2 = AppConstants.eightPoints;
1203     await firebaseClient
1204         .Child("ShoppingPoints")
1205         .Child(auth.GetUid())
1206         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, productCount = 1 });
1207
1208 }
1209 catch (NullReferenceException)
1210 {
1211     username = (await firebaseClient
1212         .Child("users")
1213         .Child(auth.GetUid())
1214         .OnceSingleAsync<Users>()).username;
1215
1216     points2 = AppConstants.eightPoints;
1217     await firebaseClient
1218         .Child("ShoppingPoints")
1219         .Child(auth.GetUid())
1220         .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, productCount = 1 });
1221 }
1222 }
1223 /**
1224 * This function updates the points in the Shopping category by four points. It
1225 * also increments the number of logs logged in the Shopping
1226 * category by one and increments the number of times this particular action was
1227 * logged by one and sends this data to Firebase.
1228 */
1229 public async void TeaPoints()
1230 {
1231
1232     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1233     quake-default-rtdb.firebaseio.com/");
1234     auth = DependencyService.Get<IAuth>();
1235
1236     try
1237     {
1238         username = (await firebaseClient
1239             .Child("users")
1240             .Child(auth.GetUid())
1241             .OnceSingleAsync<Users>()).username;
1242
1243         points2 = (await firebaseClient
1244             .Child("ShoppingPoints")
1245             .Child(auth.GetUid())
1246             .OnceSingleAsync<ShoppingPoints>()).points;
1247
1248         points2 = points2 + AppConstants.fourPoints;
1249     }
```

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

```
1306         .Child("ShoppingPoints")
1307         .Child(auth.GetUid())
1308         .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1309
1310         reBatCount2 = (await firebaseClient
1311             .Child("ShoppingPoints")
1312             .Child(auth.GetUid())
1313             .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1314
1315         reBagCount2 = (await firebaseClient
1316             .Child("ShoppingPoints")
1317             .Child(auth.GetUid())
1318             .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1319
1320         await firebaseClient
1321             .Child("ShoppingPoints")
1322             .Child(auth.GetUid())
1323             .PutAsync(new ShoppingPoints()
1324 {
1325             username = username,
1326             points = points2,
1327             numberLogs = numberLogs2,
1328             clothNapkinCount = clothNapkinCount2,
1329             clothTowelCount = clothTowelCount2,
1330             applianceCount = applianceCount2,
1331             productCount = productCount2,
1332             toothbrushCount = toothbrushCount2,
1333             clothesCount = clothesCount2,
1334             foodCount = foodCount2,
1335             localCount = localCount2,
1336             looseLeafCount = looseLeafCount2,
1337             organicFoodCount = organicFoodCount2,
1338             reusableCount = reusableCount2,
1339             reBatCount = reBatCount2,
1340             reBagCount = reBagCount2,
1341         });
1342     }
1343     catch (FirebaseException)
1344     {
1345         username = (await firebaseClient
1346             .Child("users")
1347             .Child(auth.GetUid())
1348             .OnceSingleAsync<Users>()).username;
1349
1350         points2 = AppConstants.fourPoints;
1351         await firebaseClient
1352             .Child("ShoppingPoints")
1353             .Child(auth.GetUid())
1354             .PutAsync(new ShoppingPoints() { username = username, points = points2,
1355             numberLogs = 1, looseLeafCount = 1 });
1356     }
1357     catch (NullReferenceException)
1358     {
1359         username = (await firebaseClient
1360             .Child("users")
1361             .Child(auth.GetUid())
1362             .OnceSingleAsync<Users>()).username;
1363
1364         points2 = AppConstants.fourPoints;
1365         await firebaseClient
```

```
1366             .Child("ShoppingPoints")
1367             .Child(auth.GetUid())
1368             .PutAsync(new ShoppingPoints() { username = username, points = points2,
1369             numberOfLogs = 1, looseLeafCount = 1 });
1370         }
1371         /** This function updates the points in the Shopping category by eight points. It
1372 also increments the number of logs logged in the Shopping
1373     * category by one and increments the number of times this particular action was
1374 logged by one and sends this data to Firebase.
1375 */
1376     public async void OrganicPoints()
1377     {
1378
1379         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1380         quake-default-rtdb.firebaseio.com/");
1381         auth = DependencyService.Get<IAuth>();
1382
1383         try
1384         {
1385             username = (await firebaseClient
1386             .Child("users")
1387             .Child(auth.GetUid())
1388             .OnceSingleAsync<Users>()).username;
1389
1390             points2 = (await firebaseClient
1391             .Child("ShoppingPoints")
1392             .Child(auth.GetUid())
1393             .OnceSingleAsync<ShoppingPoints>()).points;
1394
1395             points2 = points2 + AppConstants.eightPoints;
1396
1397             numberOfLogs2 = (await firebaseClient
1398             .Child("ShoppingPoints")
1399             .Child(auth.GetUid())
1400             .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
1401
1402             numberOfLogs2++;
1403
1404             clothNapkinCount2 = (await firebaseClient
1405             .Child("ShoppingPoints")
1406             .Child(auth.GetUid())
1407             .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1408
1409             clothTowelCount2 = (await firebaseClient
1410             .Child("ShoppingPoints")
1411             .Child(auth.GetUid())
1412             .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1413
1414             applianceCount2 = (await firebaseClient
1415             .Child("ShoppingPoints")
1416             .Child(auth.GetUid())
1417             .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1418
1419             productCount2 = (await firebaseClient
1420             .Child("ShoppingPoints")
1421             .Child(auth.GetUid())
1422             .OnceSingleAsync<ShoppingPoints>()).productCount;
1423
1424             toothbrushCount2 = (await firebaseClient
1425             .Child("ShoppingPoints")
1426             .Child(auth.GetUid()))
```

```
1424     .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;  
1425  
1426     clothesCount2 = (await firebaseClient  
1427         .Child("ShoppingPoints")  
1428         .Child(auth.GetUid())  
1429         .OnceSingleAsync<ShoppingPoints>()).clothesCount;  
1430  
1431     foodCount2 = (await firebaseClient  
1432         .Child("ShoppingPoints")  
1433         .Child(auth.GetUid())  
1434         .OnceSingleAsync<ShoppingPoints>()).foodCount;  
1435  
1436     localCount2 = (await firebaseClient  
1437         .Child("ShoppingPoints")  
1438         .Child(auth.GetUid())  
1439         .OnceSingleAsync<ShoppingPoints>()).localCount;  
1440  
1441     looseLeafCount2 = (await firebaseClient  
1442         .Child("ShoppingPoints")  
1443         .Child(auth.GetUid())  
1444         .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;  
1445  
1446     organicFoodCount2 = (await firebaseClient  
1447         .Child("ShoppingPoints")  
1448         .Child(auth.GetUid())  
1449         .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;  
1450  
1451     organicFoodCount2++;  
1452  
1453     reusableCount2 = (await firebaseClient  
1454         .Child("ShoppingPoints")  
1455         .Child(auth.GetUid())  
1456         .OnceSingleAsync<ShoppingPoints>()).reusableCount;  
1457  
1458     reBatCount2 = (await firebaseClient  
1459         .Child("ShoppingPoints")  
1460         .Child(auth.GetUid())  
1461         .OnceSingleAsync<ShoppingPoints>()).reBatCount;  
1462  
1463     reBagCount2 = (await firebaseClient  
1464         .Child("ShoppingPoints")  
1465         .Child(auth.GetUid())  
1466         .OnceSingleAsync<ShoppingPoints>()).reBagCount;  
1467  
1468     await firebaseClient  
1469         .Child("ShoppingPoints")  
1470         .Child(auth.GetUid())  
1471         .PutAsync(new ShoppingPoints()  
1472     {  
1473         username = username,  
1474         points = points2,  
1475         numberofLogs = numberofLogs2,  
1476         clothNapkinCount = clothNapkinCount2,  
1477         clothTowelCount = clothTowelCount2,  
1478         applianceCount = applianceCount2,  
1479         productCount = productCount2,  
1480         toothbrushCount = toothbrushCount2,  
1481         clothesCount = clothesCount2,  
1482         foodCount = foodCount2,  
1483         localCount = localCount2,  
1484         looseLeafCount = looseLeafCount2,
```

```

1485         organicFoodCount = organicFoodCount2,
1486         reusableCount = reusableCount2,
1487         reBatCount = reBatCount2,
1488         reBagCount = reBagCount2,
1489     });
1490 }
1491 catch (FirebaseException)
1492 {
1493     username = (await firebaseClient
1494     .Child("users")
1495     .Child(auth.GetUid())
1496     .OnceSingleAsync<Users>()).username;
1497
1498     points2 = AppConstants.eightPoints;
1499     await firebaseClient
1500     .Child("ShoppingPoints")
1501     .Child(auth.GetUid())
1502     .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, organicFoodCount = 1 });
1503
1504 }
1505 catch (NullReferenceException)
1506 {
1507     username = (await firebaseClient
1508     .Child("users")
1509     .Child(auth.GetUid())
1510     .OnceSingleAsync<Users>()).username;
1511
1512     points2 = AppConstants.eightPoints;
1513     await firebaseClient
1514     .Child("ShoppingPoints")
1515     .Child(auth.GetUid())
1516     .PutAsync(new ShoppingPoints() { username = username, points = points2,
numberOfLogs = 1, organicFoodCount = 1 });
1517 }
1518 }
1519 /**
1520 * This function updates the points in the Shopping category by eight points. It
1521 also increments the number of logs logged in the Shopping
1522 * category by one and increments the number of times this particular action was
1523 logged by one and sends this data to Firebase.
1524 */
1525 public async void ReWaterPoints()
1526 {
1527
1528     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1529     quake-default-firebase.firebaseio.com/");
1530     auth = DependencyService.Get<IAuth>();
1531
1532     try
1533     {
1534         username = (await firebaseClient
1535         .Child("users")
1536         .Child(auth.GetUid())
1537         .OnceSingleAsync<Users>()).username;
1538
1539         points2 = (await firebaseClient
1540         .Child("ShoppingPoints")
1541         .Child(auth.GetUid())
1542         .OnceSingleAsync<ShoppingPoints>()).points;
1543
1544         points2 = points2 + AppConstants.eightPoints;
1545     }

```

```
1542     numberLogs2 = (await firebaseClient
1543         .Child("ShoppingPoints")
1544         .Child(auth.GetUid())
1545         .OnceSingleAsync<ShoppingPoints>()).numberLogs;
1546
1547     numberLogs2++;
1548
1549     clothNapkinCount2 = (await firebaseClient
1550         .Child("ShoppingPoints")
1551         .Child(auth.GetUid())
1552         .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1553
1554     clothTowelCount2 = (await firebaseClient
1555         .Child("ShoppingPoints")
1556         .Child(auth.GetUid())
1557         .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1558
1559     applianceCount2 = (await firebaseClient
1560         .Child("ShoppingPoints")
1561         .Child(auth.GetUid())
1562         .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1563
1564     productCount2 = (await firebaseClient
1565         .Child("ShoppingPoints")
1566         .Child(auth.GetUid())
1567         .OnceSingleAsync<ShoppingPoints>()).productCount;
1568
1569     toothbrushCount2 = (await firebaseClient
1570         .Child("ShoppingPoints")
1571         .Child(auth.GetUid())
1572         .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
1573
1574     clothesCount2 = (await firebaseClient
1575         .Child("ShoppingPoints")
1576         .Child(auth.GetUid())
1577         .OnceSingleAsync<ShoppingPoints>()).clothesCount;
1578
1579     foodCount2 = (await firebaseClient
1580         .Child("ShoppingPoints")
1581         .Child(auth.GetUid())
1582         .OnceSingleAsync<ShoppingPoints>()).foodCount;
1583
1584     localCount2 = (await firebaseClient
1585         .Child("ShoppingPoints")
1586         .Child(auth.GetUid())
1587         .OnceSingleAsync<ShoppingPoints>()).localCount;
1588
1589     looseLeafCount2 = (await firebaseClient
1590         .Child("ShoppingPoints")
1591         .Child(auth.GetUid())
1592         .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1593
1594     organicFoodCount2 = (await firebaseClient
1595         .Child("ShoppingPoints")
1596         .Child(auth.GetUid())
1597         .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1598
1599     reusableCount2 = (await firebaseClient
1600         .Child("ShoppingPoints")
1601         .Child(auth.GetUid())
1602         .OnceSingleAsync<ShoppingPoints>()).reusableCount;
```

```
1603             reusableCount2++;
1604
1605
1606             reBatCount2 = (await firebaseClient
1607                 .Child("ShoppingPoints")
1608                 .Child(auth.GetUid())
1609                 .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1610
1611             reBagCount2 = (await firebaseClient
1612                 .Child("ShoppingPoints")
1613                 .Child(auth.GetUid())
1614                 .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1615
1616             await firebaseClient
1617                 .Child("ShoppingPoints")
1618                 .Child(auth.GetUid())
1619                 .PutAsync(new ShoppingPoints()
1620 {
1621             username = username,
1622             points = points2,
1623             numberLogs = numberLogs2,
1624             clothNapkinCount = clothNapkinCount2,
1625             clothTowelCount = clothTowelCount2,
1626             applianceCount = applianceCount2,
1627             productCount = productCount2,
1628             toothbrushCount = toothbrushCount2,
1629             clothesCount = clothesCount2,
1630             foodCount = foodCount2,
1631             localCount = localCount2,
1632             looseLeafCount = looseLeafCount2,
1633             organicFoodCount = organicFoodCount2,
1634             reusableCount = reusableCount2,
1635             reBatCount = reBatCount2,
1636             reBagCount = reBagCount2,
1637         });
1638     }
1639     catch (FirebaseException)
1640     {
1641         username = (await firebaseClient
1642             .Child("users")
1643             .Child(auth.GetUid())
1644             .OnceSingleAsync<Users>()).username;
1645
1646         points2 = AppConstants.eightPoints;
1647         await firebaseClient
1648             .Child("ShoppingPoints")
1649             .Child(auth.GetUid())
1650             .PutAsync(new ShoppingPoints() { username = username, points = points2,
1651             numberLogs = 1, reusableCount = 1 });
1652     }
1653     catch (NullReferenceException)
1654     {
1655         username = (await firebaseClient
1656             .Child("users")
1657             .Child(auth.GetUid())
1658             .OnceSingleAsync<Users>()).username;
1659
1660         points2 = AppConstants.eightPoints;
1661         await firebaseClient
1662             .Child("ShoppingPoints")
```

```
1663         .Child(auth.GetUid())
1664         .PutAsync(new ShoppingPoints() { username = username, points = points2,
1665 numberOfLogs = 1, reusableCount = 1 });
1666     }
1667     /** This function updates the points in the Shopping category by six points. It
1668 also increments the number of logs logged in the Shopping
1669     * category by one and increments the number of times this particular action was
1670 logged by one and sends this data to Firebase.
1671     */
1672     public async void ReBatttereisPoints()
1673     {
1674
1675         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1676 quake-default-rtdb.firebaseio.com/");
1677         auth = DependencyService.Get<IAuth>();
1678
1679         try
1680         {
1681             username = (await firebaseClient
1682                 .Child("users")
1683                 .Child(auth.GetUid())
1684                 .OnceSingleAsync<Users>()).username;
1685
1686             points2 = (await firebaseClient
1687                 .Child("ShoppingPoints")
1688                 .Child(auth.GetUid())
1689                 .OnceSingleAsync<ShoppingPoints>()).points;
1690
1691             points2 = points2 + AppConstants.sixPoints;
1692
1693             numberOfLogs2 = (await firebaseClient
1694                 .Child("ShoppingPoints")
1695                 .Child(auth.GetUid())
1696                 .OnceSingleAsync<ShoppingPoints>()).numberOfLogs;
1697
1698             numberOfLogs2++;
1699
1700             clothNapkinCount2 = (await firebaseClient
1701                 .Child("ShoppingPoints")
1702                 .Child(auth.GetUid())
1703                 .OnceSingleAsync<ShoppingPoints>()).clothNapkinCount;
1704
1705             clothTowelCount2 = (await firebaseClient
1706                 .Child("ShoppingPoints")
1707                 .Child(auth.GetUid())
1708                 .OnceSingleAsync<ShoppingPoints>()).clothTowelCount;
1709
1710             applianceCount2 = (await firebaseClient
1711                 .Child("ShoppingPoints")
1712                 .Child(auth.GetUid())
1713                 .OnceSingleAsync<ShoppingPoints>()).applianceCount;
1714
1715             productCount2 = (await firebaseClient
1716                 .Child("ShoppingPoints")
1717                 .Child(auth.GetUid())
1718                 .OnceSingleAsync<ShoppingPoints>()).productCount;
1719
1720             toothbrushCount2 = (await firebaseClient
1721                 .Child("ShoppingPoints")
1722                 .Child(auth.GetUid())
1723                 .OnceSingleAsync<ShoppingPoints>()).toothbrushCount;
```

```
1721
1722     clothesCount2 = (await firebaseClient
1723         .Child("ShoppingPoints")
1724         .Child(auth.GetUid())
1725         .OnceSingleAsync<ShoppingPoints>()).clothesCount;
1726
1727     foodCount2 = (await firebaseClient
1728         .Child("ShoppingPoints")
1729         .Child(auth.GetUid())
1730         .OnceSingleAsync<ShoppingPoints>()).foodCount;
1731
1732     localCount2 = (await firebaseClient
1733         .Child("ShoppingPoints")
1734         .Child(auth.GetUid())
1735         .OnceSingleAsync<ShoppingPoints>()).localCount;
1736
1737     looseLeafCount2 = (await firebaseClient
1738         .Child("ShoppingPoints")
1739         .Child(auth.GetUid())
1740         .OnceSingleAsync<ShoppingPoints>()).looseLeafCount;
1741
1742     organicFoodCount2 = (await firebaseClient
1743         .Child("ShoppingPoints")
1744         .Child(auth.GetUid())
1745         .OnceSingleAsync<ShoppingPoints>()).organicFoodCount;
1746
1747     reusableCount2 = (await firebaseClient
1748         .Child("ShoppingPoints")
1749         .Child(auth.GetUid())
1750         .OnceSingleAsync<ShoppingPoints>()).reusableCount;
1751
1752     reBatCount2 = (await firebaseClient
1753         .Child("ShoppingPoints")
1754         .Child(auth.GetUid())
1755         .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1756
1757     reBatCount2++;
1758
1759     reBagCount2 = (await firebaseClient
1760         .Child("ShoppingPoints")
1761         .Child(auth.GetUid())
1762         .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1763
1764     await firebaseClient
1765         .Child("ShoppingPoints")
1766         .Child(auth.GetUid())
1767         .PutAsync(new ShoppingPoints()
1768     {
1769         username = username,
1770         points = points2,
1771         numberofLogs = numberofLogs2,
1772         clothNapkinCount = clothNapkinCount2,
1773         clothTowelCount = clothTowelCount2,
1774         applianceCount = applianceCount2,
1775         productCount = productCount2,
1776         toothbrushCount = toothbrushCount2,
1777         clothesCount = clothesCount2,
1778         foodCount = foodCount2,
1779         localCount = localCount2,
1780         looseLeafCount = looseLeafCount2,
1781         organicFoodCount = organicFoodCount2,
```

```

1782             reusableCount = reusableCount2,
1783             reBatCount = reBatCount2,
1784             reBagCount = reBagCount2,
1785         });
1786     }
1787     catch (FirebaseException)
1788     {
1789         username = (await firebaseClient
1790             .Child("users")
1791             .Child(auth.GetUid())
1792             .OnceSingleAsync<Users>()).username;
1793
1794         points2 = AppConstants.sixPoints;
1795         await firebaseClient
1796             .Child("ShoppingPoints")
1797             .Child(auth.GetUid())
1798             .PutAsync(new ShoppingPoints() { username = username, points = points2,
1799             numberOfLogs = 1, reBatCount = 1 });
1800     }
1801     catch (NullReferenceException)
1802     {
1803         username = (await firebaseClient
1804             .Child("users")
1805             .Child(auth.GetUid())
1806             .OnceSingleAsync<Users>()).username;
1807
1808         points2 = AppConstants.sixPoints;
1809         await firebaseClient
1810             .Child("ShoppingPoints")
1811             .Child(auth.GetUid())
1812             .PutAsync(new ShoppingPoints() { username = username, points = points2,
1813             numberOfLogs = 1, reBatCount = 1 });
1814     }
1815     /** This function updates the points in the Shopping category by eight points. It
1816 also increments the number of logs logged in the Shopping
1817     * category by one and increments the number of times this particular action was
1818     * logged by one and sends this data to Firebase.
1819 */
1820     public async void ReBagPoints()
1821     {
1822
1823         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
1824         quake-default-rtdb.firebaseio.com/");
1825         auth = DependencyService.Get<IAuth>();
1826
1827         try
1828         {
1829             username = (await firebaseClient
1830                 .Child("users")
1831                 .Child(auth.GetUid())
1832                 .OnceSingleAsync<Users>()).username;
1833
1834             points2 = (await firebaseClient
1835                 .Child("ShoppingPoints")
1836                 .Child(auth.GetUid())
1837                 .OnceSingleAsync<ShoppingPoints>()).points;
1838
1839             points2 = points2 + AppConstants.eightPoints;
1840
1841             numberOfLogs2 = (await firebaseClient

```

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

```

1900    reBatCount2 = (await firebaseClient
1901        .Child("ShoppingPoints")
1902        .Child(auth.GetUid())
1903        .OnceSingleAsync<ShoppingPoints>()).reBatCount;
1904
1905    reBagCount2 = (await firebaseClient
1906        .Child("ShoppingPoints")
1907        .Child(auth.GetUid())
1908        .OnceSingleAsync<ShoppingPoints>()).reBagCount;
1909
1910    reBagCount2++;
1911
1912    await firebaseClient
1913        .Child("ShoppingPoints")
1914        .Child(auth.GetUid())
1915        .PutAsync(new ShoppingPoints()
1916    {
1917        username = username,
1918        points = points2,
1919        numberLogs = numberLogs2,
1920        clothNapkinCount = clothNapkinCount2,
1921        clothTowelCount = clothTowelCount2,
1922        applianceCount = applianceCount2,
1923        productCount = productCount2,
1924        toothbrushCount = toothbrushCount2,
1925        clothesCount = clothesCount2,
1926        foodCount = foodCount2,
1927        localCount = localCount2,
1928        looseLeafCount = looseLeafCount2,
1929        organicFoodCount = organicFoodCount2,
1930        reusableCount = reusableCount2,
1931        reBatCount = reBatCount2,
1932        reBagCount = reBagCount2,
1933    });
1934 }
1935 catch (FirebaseException)
1936 {
1937     username = (await firebaseClient
1938         .Child("users")
1939         .Child(auth.GetUid())
1940         .OnceSingleAsync<Users>()).username;
1941
1942     points2 = AppConstants.eightPoints;
1943     await firebaseClient
1944         .Child("ShoppingPoints")
1945         .Child(auth.GetUid())
1946         .PutAsync(new ShoppingPoints() { username = username, points = points2,
1947     numberLogs = 1, reBagCount = 1 });
1948 }
1949 catch (NullReferenceException)
1950 {
1951     username = (await firebaseClient
1952         .Child("users")
1953         .Child(auth.GetUid())
1954         .OnceSingleAsync<Users>()).username;
1955
1956     points2 = AppConstants.eightPoints;
1957     await firebaseClient
1958         .Child("ShoppingPoints")
1959         .Child(auth.GetUid())

```

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

```
1 /*! \class The TravelPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the TravelPointsUpdate ViewModel Class. It updates the data for the
8  * Travel Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 * this data to back firebase.
11 */
12
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class TravelPointsUpdate
22     {
23         int points2 = 0;
24         int number0fLogs2 = 0;
25         int carpoolCount2 = 0;
26         int cycleCount2 = 0;
27         int ecoCarCount2 = 0;
28         int transportCount2 = 0;
29         int walkCount2 = 0;
30
31         string username = "";
32
33         IAuth auth;
34         /** This function updates the points in the Travel category by six points. It also
35          increments the number of logs logged in the Travel
36          * category by one and increments the number of times this particular action was
37          logged by one and sends this data to Firebase.
38         */
39         public async void CarpoolPoints()
40         {
41
42             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
43             quake-default-rtdb.firebaseio.com/");
44             auth = DependencyService.Get<IAuth>();
45
46             try
47             {
48                 username = (await firebaseClient
49                     .Child("users")
50                     .Child(auth.GetUid())
51                     .OnceSingleAsync<Users>()).username;
52
53                 points2 = (await firebaseClient
54                     .Child("TravelPoints")
55                     .Child(auth.GetUid())
56                     .OnceSingleAsync<TravelPoints>()).points;
57
58                 points2 = points2 + AppConstants.sixPoints;
59
60                 number0fLogs2 = (await firebaseClient
61                     .Child("TravelPoints")
```

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

```
117     }
118     catch (NullReferenceException)
119     {
120         username = (await firebaseClient
121             .Child("users")
122             .Child(auth.GetUid())
123             .OnceSingleAsync<Users>()).username;
124
125         points2 = AppConstants.sixPoints;
126         await firebaseClient
127             .Child("TravelPoints")
128             .Child(auth.GetUid())
129             .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, carpoolCount = 1 });
130     }
131 }
132 /**
133  * This function updates the points in the Travel category by ten points. It also
134 increments the number of logs logged in the Travel
135 * category by one and increments the number of times this particular action was
136 logged by one and sends this data to Firebase.
137 */
138 public async void CyclePoints()
139 {
140
141     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
142     quake-default-rtdb.firebaseio.com/");
143     auth = DependencyService.Get<IAuth>();
144
145     try
146     {
147         username = (await firebaseClient
148             .Child("users")
149             .Child(auth.GetUid())
150             .OnceSingleAsync<Users>()).username;
151
152         points2 = (await firebaseClient
153             .Child("TravelPoints")
154             .Child(auth.GetUid())
155             .OnceSingleAsync<TravelPoints>()).points;
156
157         points2 = points2 + AppConstants.tenPoints;
158
159         numberOfLogs2 = (await firebaseClient
160             .Child("TravelPoints")
161             .Child(auth.GetUid())
162             .OnceSingleAsync<TravelPoints>()).numberOfLogs;
163
164         numberOfLogs2++;
165
166         carpoolCount2 = (await firebaseClient
167             .Child("TravelPoints")
168             .Child(auth.GetUid())
169             .OnceSingleAsync<TravelPoints>()).carpoolCount;
170
171         cycleCount2 = (await firebaseClient
172             .Child("TravelPoints")
173             .Child(auth.GetUid())
174             .OnceSingleAsync<TravelPoints>()).cycleCount;
175
176         cycleCount2++;
177
178         ecoCarCount2 = (await firebaseClient
```

```

175     .Child("TravelPoints")
176     .Child(auth.GetUid())
177     .OnceSingleAsync<TravelPoints>()).ecoCarCount;
178
179     transportCount2 = (await firebaseClient
180     .Child("TravelPoints")
181     .Child(auth.GetUid())
182     .OnceSingleAsync<TravelPoints>()).transportCount;
183
184     walkCount2 = (await firebaseClient
185     .Child("TravelPoints")
186     .Child(auth.GetUid())
187     .OnceSingleAsync<TravelPoints>()).walkCount;
188
189     await firebaseClient
190     .Child("TravelPoints")
191     .Child(auth.GetUid())
192     .PutAsync(new TravelPoints()
193     {
194         username = username,
195         points = points2,
196         numberOfLogs = numberOfLogs2,
197         carpoolCount = carpoolCount2,
198         cycleCount = cycleCount2,
199         ecoCarCount = ecoCarCount2,
200         transportCount = transportCount2,
201         walkCount = walkCount2,
202     });
203 }
204 catch (FirebaseException)
205 {
206     username = (await firebaseClient
207     .Child("users")
208     .Child(auth.GetUid())
209     .OnceSingleAsync<Users>()).username;
210
211     points2 = AppConstants.tenPoints;
212     await firebaseClient
213     .Child("TravelPoints")
214     .Child(auth.GetUid())
215     .PutAsync(new TravelPoints() { username = username, points = points2,
216     numberOfLogs = 1, cycleCount = 1 });
217
218     catch (NullReferenceException)
219     {
220         username = (await firebaseClient
221         .Child("users")
222         .Child(auth.GetUid())
223         .OnceSingleAsync<Users>()).username;
224
225         points2 = AppConstants.tenPoints;
226         await firebaseClient
227         .Child("TravelPoints")
228         .Child(auth.GetUid())
229         .PutAsync(new TravelPoints() { username = username, points = points2,
230     numberOfLogs = 1, cycleCount = 1 });
231     }
232     /** This function updates the points in the Travel category by ten points. It also
233     increments the number of logs logged in the Travel
234     * category by one and increments the number of times this particular action was

```

```
logged by one and sends this data to Firebase.  
234     */  
235     public async void EcoCarPoints()  
236     {  
237  
238         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-  
quake-default-rtdb.firebaseio.com/");  
239         auth = DependencyService.Get<IAuth>();  
240  
241         try  
242         {  
243             username = (await firebaseClient  
244                 .Child("users")  
245                 .Child(auth.GetUid())  
246                 .OnceSingleAsync<Users>()).username;  
247  
248             points2 = (await firebaseClient  
249                 .Child("TravelPoints")  
250                 .Child(auth.GetUid())  
251                 .OnceSingleAsync<TravelPoints>()).points;  
252  
253             points2 = points2 + AppConstants.tenPoints;  
254  
255             numberofLogs2 = (await firebaseClient  
256                 .Child("TravelPoints")  
257                 .Child(auth.GetUid())  
258                 .OnceSingleAsync<TravelPoints>()).numberofLogs;  
259  
260             numberofLogs2++;  
261  
262             carpoolCount2 = (await firebaseClient  
263                 .Child("TravelPoints")  
264                 .Child(auth.GetUid())  
265                 .OnceSingleAsync<TravelPoints>()).carpoolCount;  
266  
267             cycleCount2 = (await firebaseClient  
268                 .Child("TravelPoints")  
269                 .Child(auth.GetUid())  
270                 .OnceSingleAsync<TravelPoints>()).cycleCount;  
271  
272             ecoCarCount2 = (await firebaseClient  
273                 .Child("TravelPoints")  
274                 .Child(auth.GetUid())  
275                 .OnceSingleAsync<TravelPoints>()).ecoCarCount;  
276  
277             ecoCarCount2++;  
278  
279             transportCount2 = (await firebaseClient  
280                 .Child("TravelPoints")  
281                 .Child(auth.GetUid())  
282                 .OnceSingleAsync<TravelPoints>()).transportCount;  
283  
284             walkCount2 = (await firebaseClient  
285                 .Child("TravelPoints")  
286                 .Child(auth.GetUid())  
287                 .OnceSingleAsync<TravelPoints>()).walkCount;  
288  
289             await firebaseClient  
290                 .Child("TravelPoints")  
291                 .Child(auth.GetUid())  
292                 .PutAsync(new TravelPoints()
```

```

293     {
294         username = username,
295         points = points2,
296         numberOfLogs = numberOfLogs2,
297         carpoolCount = carpoolCount2,
298         cycleCount = cycleCount2,
299         ecoCarCount = ecoCarCount2,
300         transportCount = transportCount2,
301         walkCount = walkCount2,
302     });
303 }
304 catch (FirebaseException)
305 {
306     username = (await firebaseClient
307     .Child("users")
308     .Child(auth.GetUid())
309     .OnceSingleAsync<Users>()).username;
310
311     points2 = AppConstants.tenPoints;
312     await firebaseClient
313     .Child("TravelPoints")
314     .Child(auth.GetUid())
315     .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, ecoCarCount = 1 });
316 }
317 catch (NullReferenceException)
318 {
319     username = (await firebaseClient
320     .Child("users")
321     .Child(auth.GetUid())
322     .OnceSingleAsync<Users>()).username;
323
324     points2 = AppConstants.tenPoints;
325     await firebaseClient
326     .Child("TravelPoints")
327     .Child(auth.GetUid())
328     .PutAsync(new TravelPoints() { username = username, points = points2,
numberOfLogs = 1, ecoCarCount = 1 });
329 }
330 */
331 /**
332  * This function updates the points in the Travel category by eight points. It
333  * also increments the number of logs logged in the Travel
334  * category by one and increments the number of times this particular action was
335  * logged by one and sends this data to Firebase.
336 */
337 public async void TransportPoints()
338 {
339     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
340     quake-default-rtdb.firebaseio.com/");
341     auth = DependencyService.Get<IAuth>();
342
343     try
344     {
345         username = (await firebaseClient
346         .Child("users")
347         .Child(auth.GetUid())
348         .OnceSingleAsync<Users>()).username;
349
350         points2 = (await firebaseClient
351         .Child("TravelPoints")
352

```

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

```
411     points2 = AppConstants.eightPoints;
412     await firebaseClient
413         .Child("TravelPoints")
414         .Child(auth.GetUid())
415         .PutAsync(new TravelPoints() { username = username, points = points2,
416     numberOfLogs = 1, transportCount = 1 });
417 }
418 catch (NullReferenceException)
419 {
420     username = (await firebaseClient
421         .Child("users")
422         .Child(auth.GetUid())
423         .OnceSingleAsync<Users>()).username;
424
425     points2 = AppConstants.eightPoints;
426     await firebaseClient
427         .Child("TravelPoints")
428         .Child(auth.GetUid())
429         .PutAsync(new TravelPoints() { username = username, points = points2,
430     numberOfLogs = 1, transportCount = 1 });
431 }
432 /**
433 * This function updates the points in the Travel category by ten points. It also
434 increments the number of logs logged in the Travel
435 * category by one and increments the number of times this particular action was
436 logged by one and sends this data to Firebase.
437 */
438 public async void WalkPoints()
439 {
440
441     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
442     quake-default-rtdb.firebaseio.com/");
443     auth = DependencyService.Get<IAuth>();
444
445     try
446     {
447         username = (await firebaseClient
448             .Child("users")
449             .Child(auth.GetUid())
450             .OnceSingleAsync<Users>()).username;
451
452         points2 = (await firebaseClient
453             .Child("TravelPoints")
454             .Child(auth.GetUid())
455             .OnceSingleAsync<TravelPoints>()).points;
456
457         points2 = points2 + AppConstants.tenPoints;
458
459         numberOfLogs2 = (await firebaseClient
460             .Child("TravelPoints")
461             .Child(auth.GetUid())
462             .OnceSingleAsync<TravelPoints>()).numberOfLogs;
463
464         numberOfLogs2++;
465
466         carpoolCount2 = (await firebaseClient
467             .Child("TravelPoints")
468             .Child(auth.GetUid())
469             .OnceSingleAsync<TravelPoints>()).carpoolCount;
470
471         cycleCount2 = (await firebaseClient
```

```
468     .Child("TravelPoints")
469     .Child(auth.GetUid())
470     .OnceSingleAsync<TravelPoints>()).cycleCount;
471 
472     ecoCarCount2 = (await firebaseClient
473     .Child("TravelPoints")
474     .Child(auth.GetUid())
475     .OnceSingleAsync<TravelPoints>()).ecoCarCount;
476 
477     transportCount2 = (await firebaseClient
478     .Child("TravelPoints")
479     .Child(auth.GetUid())
480     .OnceSingleAsync<TravelPoints>()).transportCount;
481 
482     walkCount2 = (await firebaseClient
483     .Child("TravelPoints")
484     .Child(auth.GetUid())
485     .OnceSingleAsync<TravelPoints>()).walkCount;
486 
487     walkCount2++;
488 
489     await firebaseClient
490     .Child("TravelPoints")
491     .Child(auth.GetUid())
492     .PutAsync(new TravelPoints())
493     {
494         username = username,
495         points = points2,
496         numberofLogs = numberofLogs2,
497         carpoolCount = carpoolCount2,
498         cycleCount = cycleCount2,
499         ecoCarCount = ecoCarCount2,
500         transportCount = transportCount2,
501         walkCount = walkCount2,
502     });
503 }
504 catch (FirebaseException)
505 {
506     username = (await firebaseClient
507     .Child("users")
508     .Child(auth.GetUid())
509     .OnceSingleAsync<Users>()).username;
510 
511     points2 = AppConstants.tenPoints;
512     await firebaseClient
513     .Child("TravelPoints")
514     .Child(auth.GetUid())
515     .PutAsync(new TravelPoints() { username = username, points = points2,
516     numberofLogs = 1, walkCount = 1 });
517 }
518 catch (NullReferenceException)
519 {
520     username = (await firebaseClient
521     .Child("users")
522     .Child(auth.GetUid())
523     .OnceSingleAsync<Users>()).username;
524 
525     points2 = AppConstants.tenPoints;
526     await firebaseClient
527     .Child("TravelPoints")
```

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

```
1 /*! \class The WastePointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the WastePointsUpdate ViewModel Class. It updates the data for the
8  * Waste Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 * this data to back firebase.
11 */
12
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class WastePointsUpdate
22     {
23         int points2 = 0;
24         int numberLogs2 = 0;
25         int billsCount2 = 0;
26         int compostCount2 = 0;
27         int setUpRecyclingBinCount2 = 0;
28         int bioBinBagsCount2 = 0;
29         int recyclingBinCount2 = 0;
30
31         string username = "";
32
33         IAuth auth;
34         /** This function updates the points in the Waste category by four points. It also
35          increments the number of logs logged in the Waste
36          * category by one and increments the number of times this particular action was
37          logged by one and sends this data to Firebase.
38
39         */
40         public async void BillsPoints()
41         {
42
43             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
44             quake-default-rtdb.firebaseio.com/");
45             auth = DependencyService.Get<IAuth>();
46
47             try
48             {
49                 username = (await firebaseClient
50                     .Child("users")
51                     .Child(auth.GetUid())
52                     .OnceSingleAsync<Users>()).username;
53
54                 points2 = (await firebaseClient
55                     .Child("WastePoints")
56                     .Child(auth.GetUid())
57                     .OnceSingleAsync<WastePoints>()).points;
58
59                 points2 = points2 + AppConstants.fourPoints;
60
61                 numberLogs2 = (await firebaseClient
62                     .Child("WastePoints")
```

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

```
117     }
118     catch (NullReferenceException)
119     {
120         username = (await firebaseClient
121             .Child("users")
122             .Child(auth.GetUid())
123             .OnceSingleAsync<Users>()).username;
124
125         points2 = AppConstants.fourPoints;
126         await firebaseClient
127             .Child("WastePoints")
128             .Child(auth.GetUid())
129             .PutAsync(new WastePoints() { username = username, points = points2,
numberOfLogs = 1, billsCount = 1 });
130     }
131 }
132 /**
133  * This function updates the points in the Waste category by six points. It also
134 increments the number of logs logged in the Waste
135 * category by one and increments the number of times this particular action was
136 logged by one and sends this data to Firebase.
137 */
138 public async void CompostPoints()
139 {
140
141     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
142     quake-default-rtdb.firebaseio.com/");
143     auth = DependencyService.Get<IAuth>();
144
145     try
146     {
147         username = (await firebaseClient
148             .Child("users")
149             .Child(auth.GetUid())
150             .OnceSingleAsync<Users>()).username;
151
152         points2 = (await firebaseClient
153             .Child("WastePoints")
154             .Child(auth.GetUid())
155             .OnceSingleAsync<WastePoints>()).points;
156
157         points2 = points2 + AppConstants.sixPoints;
158
159         numberOfLogs2 = (await firebaseClient
160             .Child("WastePoints")
161             .Child(auth.GetUid())
162             .OnceSingleAsync<WastePoints>()).numberOfLogs;
163
164         numberOfLogs2++;
165
166         billsCount2 = (await firebaseClient
167             .Child("WastePoints")
168             .Child(auth.GetUid())
169             .OnceSingleAsync<WastePoints>()).billsCount;
170
171         compostCount2 = (await firebaseClient
172             .Child("WastePoints")
173             .Child(auth.GetUid())
174             .OnceSingleAsync<WastePoints>()).compostCount;
175
176         compostCount2++;
177
178         setUpRecyclingBinCount2 = (await firebaseClient
```

```

175     .Child("WastePoints")
176     .Child(auth.GetUid())
177     .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
178
179     bioBinBagsCount2 = (await firebaseClient
180     .Child("WastePoints")
181     .Child(auth.GetUid())
182     .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
183
184     recyclingBinCount2 = (await firebaseClient
185     .Child("WastePoints")
186     .Child(auth.GetUid())
187     .OnceSingleAsync<WastePoints>()).recyclingBinCount;
188
189     await firebaseClient
190     .Child("WastePoints")
191     .Child(auth.GetUid())
192     .PutAsync(new WastePoints())
193     {
194         username = username,
195         points = points2,
196         numberOfLogs = numberOfLogs2,
197         billsCount = billsCount2,
198         compostCount = compostCount2,
199         setUpRecyclingBinCount = setUpRecyclingBinCount2,
200         bioBinBagsCount = bioBinBagsCount2,
201         recyclingBinCount = recyclingBinCount2,
202     });
203 }
204 catch (FirebaseException)
205 {
206     username = (await firebaseClient
207     .Child("users")
208     .Child(auth.GetUid())
209     .OnceSingleAsync<Users>()).username;
210
211     points2 = AppConstants.sixPoints;
212     await firebaseClient
213     .Child("WastePoints")
214     .Child(auth.GetUid())
215     .PutAsync(new WastePoints() { username = username, points = points2,
216     numberOfLogs = 1, compostCount = 1 });
217
218     catch (NullReferenceException)
219     {
220         username = (await firebaseClient
221         .Child("users")
222         .Child(auth.GetUid())
223         .OnceSingleAsync<Users>()).username;
224
225         points2 = AppConstants.sixPoints;
226         await firebaseClient
227         .Child("WastePoints")
228         .Child(auth.GetUid())
229         .PutAsync(new WastePoints() { username = username, points = points2,
230     numberOfLogs = 1, compostCount = 1 });
231     }
232     /** This function updates the points in the Waste category by ten points. It also
233     increments the number of logs logged in the Waste
     * category by one and increments the number of times this particular action was

```

```
logged by one and sends this data to Firebase.  
234     */  
235     public async void SetUpRecyclingBinPoints()  
236     {  
237  
238         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-  
quake-default-firebase.com.firebaseio.com/");  
239         auth = DependencyService.Get<IAuth>();  
240  
241         try  
242         {  
243             username = (await firebaseClient  
244                 .Child("users")  
245                 .Child(auth.GetUid())  
246                 .OnceSingleAsync<Users>()).username;  
247  
248             points2 = (await firebaseClient  
249                 .Child("WastePoints")  
250                 .Child(auth.GetUid())  
251                 .OnceSingleAsync<WastePoints>()).points;  
252  
253             points2 = points2 + AppConstants.tenPoints;  
254  
255             numberofLogs2 = (await firebaseClient  
256                 .Child("WastePoints")  
257                 .Child(auth.GetUid())  
258                 .OnceSingleAsync<WastePoints>()).numberofLogs;  
259  
260             numberofLogs2++;  
261  
262             billsCount2 = (await firebaseClient  
263                 .Child("WastePoints")  
264                 .Child(auth.GetUid())  
265                 .OnceSingleAsync<WastePoints>()).billsCount;  
266  
267             compostCount2 = (await firebaseClient  
268                 .Child("WastePoints")  
269                 .Child(auth.GetUid())  
270                 .OnceSingleAsync<WastePoints>()).compostCount;  
271  
272             setUpRecyclingBinCount2 = (await firebaseClient  
273                 .Child("WastePoints")  
274                 .Child(auth.GetUid())  
275                 .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;  
276  
277             setUpRecyclingBinCount2++;  
278  
279             bioBinBagsCount2 = (await firebaseClient  
280                 .Child("WastePoints")  
281                 .Child(auth.GetUid())  
282                 .OnceSingleAsync<WastePoints>()).bioBinBagsCount;  
283  
284             recyclingBinCount2 = (await firebaseClient  
285                 .Child("WastePoints")  
286                 .Child(auth.GetUid())  
287                 .OnceSingleAsync<WastePoints>()).recyclingBinCount;  
288  
289             await firebaseClient  
290                 .Child("WastePoints")  
291                 .Child(auth.GetUid())  
292                 .PutAsync(new WastePoints()
```

```

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

```

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

```

411     points2 = AppConstants.fourPoints;
412     await firebaseClient
413         .Child("WastePoints")
414         .Child(auth.GetUid())
415         .PutAsync(new WastePoints() { username = username, points = points2,
416     numberOfLogs = 1, bioBinBagsCount = 1 });
417 }
418 catch (NullReferenceException)
419 {
420     username = (await firebaseClient
421         .Child("users")
422         .Child(auth.GetUid())
423         .OnceSingleAsync<Users>()).username;
424
425     points2 = AppConstants.fourPoints;
426     await firebaseClient
427         .Child("WastePoints")
428         .Child(auth.GetUid())
429         .PutAsync(new WastePoints() { username = username, points = points2,
430     numberOfLogs = 1, bioBinBagsCount = 1 });
431 }
432 /**
433 * This function updates the points in the Waste category by six points. It also
434 increments the number of logs logged in the Waste
435 * category by one and increments the number of times this particular action was
436 logged by one and sends this data to Firebase.
437 */
438 public async void RecyclingBinPoints()
439 {
440
441     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
442     quake-default-rtdb.firebaseio.com/");
443     auth = DependencyService.Get<IAuth>();
444
445     try
446     {
447         username = (await firebaseClient
448             .Child("users")
449             .Child(auth.GetUid())
450             .OnceSingleAsync<Users>()).username;
451
452         points2 = (await firebaseClient
453             .Child("WastePoints")
454             .Child(auth.GetUid())
455             .OnceSingleAsync<WastePoints>()).points;
456
457         points2 = points2 + AppConstants.sixPoints;
458
459         numberOfLogs2 = (await firebaseClient
460             .Child("WastePoints")
461             .Child(auth.GetUid())
462             .OnceSingleAsync<WastePoints>()).numberOfLogs;
463
464         numberOfLogs2++;
465
466         billsCount2 = (await firebaseClient
467             .Child("WastePoints")
468             .Child(auth.GetUid())
469             .OnceSingleAsync<WastePoints>()).billsCount;
470
471         compostCount2 = (await firebaseClient

```

```
468     .Child("WastePoints")
469     .Child(auth.GetUid())
470     .OnceSingleAsync<WastePoints>()).compostCount;
471 
472     setUpRecyclingBinCount2 = (await firebaseClient
473     .Child("WastePoints")
474     .Child(auth.GetUid())
475     .OnceSingleAsync<WastePoints>()).setUpRecyclingBinCount;
476 
477     bioBinBagsCount2 = (await firebaseClient
478     .Child("WastePoints")
479     .Child(auth.GetUid())
480     .OnceSingleAsync<WastePoints>()).bioBinBagsCount;
481 
482     recyclingBinCount2 = (await firebaseClient
483     .Child("WastePoints")
484     .Child(auth.GetUid())
485     .OnceSingleAsync<WastePoints>()).recyclingBinCount;
486 
487     recyclingBinCount2++;
488 
489     await firebaseClient
490     .Child("WastePoints")
491     .Child(auth.GetUid())
492     .PutAsync(new WastePoints())
493     {
494         username = username,
495         points = points2,
496         numberofLogs = numberofLogs2,
497         billsCount = billsCount2,
498         compostCount = compostCount2,
499         setUpRecyclingBinCount = setUpRecyclingBinCount2,
500         bioBinBagsCount = bioBinBagsCount2,
501         recyclingBinCount = recyclingBinCount2,
502     });
503 }
504 catch (FirebaseException)
505 {
506     username = (await firebaseClient
507     .Child("users")
508     .Child(auth.GetUid())
509     .OnceSingleAsync<Users>()).username;
510 
511     points2 = AppConstants.sixPoints;
512     await firebaseClient
513     .Child("WastePoints")
514     .Child(auth.GetUid())
515     .PutAsync(new WastePoints() { username = username, points = points2,
516     numberofLogs = 1, recyclingBinCount = 1 });
517 }
518 catch (NullReferenceException)
519 {
520     username = (await firebaseClient
521     .Child("users")
522     .Child(auth.GetUid())
523     .OnceSingleAsync<Users>()).username;
524 
525     points2 = AppConstants.sixPoints;
526     await firebaseClient
527     .Child("WastePoints")
```

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

```
1 /*! \class The WaterPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the WaterPointsUpdate ViewModel Class. It updates the data for the
8  * Water Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 this data to back firebase.
11 *
12 */
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class WaterPointsUpdate
22     {
23         int points2 = 0;
24         int numberLogs2 = 0;
25         int cisternCount2 = 0;
26         int rainBarrel2 = 0;
27         int reWater2 = 0;
28         int showerBucket2 = 0;
29         int wSShowerHead2 = 0;
30
31         string username = "";
32
33         IAuth auth;
34         /** This function updates the points in the Water category by ten points. It also
35 increments the number of logs logged in the Water
36      * category by one and increments the number of times this particular action was
37      * logged by one and sends this data to Firebase.
38      */
39 public async void CisternPoints()
40     {
41
42         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
43         quake-default-rtdb.firebaseio.com/");
44         auth = DependencyService.Get<IAuth>();
45
46         try
47         {
48             username = (await firebaseClient
49             .Child("users")
50             .Child(auth.GetUid())
51             .OnceSingleAsync<Users>()).username;
52
53             points2 = (await firebaseClient
54             .Child("WaterPoints")
55             .Child(auth.GetUid())
56             .OnceSingleAsync<WaterPoints>()).points;
57
58             points2 = points2 + AppConstants.tenPoints;
59
60             numberLogs2 = (await firebaseClient
61             .Child("WaterPoints")
62             .Child(auth.GetUid())
63             .OnceSingleAsync<WaterPoints>()).points;
64
65             numberLogs2 = numberLogs2 + 1;
66
67             var waterPointsRef = firebaseClient
68             .Child("WaterPoints")
69             .Child(auth.GetUid());
70
71             waterPointsRef
72             .Child("points")
73             .Value = points2;
74
75             waterPointsRef
76             .Child("logs")
77             .Value = numberLogs2;
78
79             waterPointsRef
80             .Child("cisternCount")
81             .Value = cisternCount2;
82
83             waterPointsRef
84             .Child("rainBarrel")
85             .Value = rainBarrel2;
86
87             waterPointsRef
88             .Child("reWater")
89             .Value = reWater2;
90
91             waterPointsRef
92             .Child("showerBucket")
93             .Value = showerBucket2;
94
95             waterPointsRef
96             .Child("wSShowerHead")
97             .Value = wSShowerHead2;
98
99             return Task.CompletedTask;
100     }
101 }
```

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

```
117     catch (NullReferenceException)
118     {
119         username = (await firebaseClient
120             .Child("users")
121             .Child(auth.GetUid())
122             .OnceSingleAsync<Users>()).username;
123
124         points2 = AppConstants.tenPoints;
125         await firebaseClient
126             .Child("WaterPoints")
127             .Child(auth.GetUid())
128             .PutAsync(new WaterPoints() { username = username, points = points2,
129             numberOfLogs = 1, cisternCount = 1 });
130     }
131     /** This function updates the points in the Water category by ten points. It also
132     increments the number of logs logged in the Water
133     * category by one and increments the number of times this particular action was
134     logged by one and sends this data to Firebase.
135     */
136     public async void BarrelPoints()
137     {
138
139         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
140         quake-default-rtdb.firebaseio.com/");
141         auth = DependencyService.Get<IAuth>();
142
143         try
144         {
145             username = (await firebaseClient
146                 .Child("users")
147                     .Child(auth.GetUid())
148                     .OnceSingleAsync<Users>()).username;
149
150             points2 = (await firebaseClient
151                 .Child("WaterPoints")
152                     .Child(auth.GetUid())
153                     .OnceSingleAsync<WaterPoints>()).points;
154
155             points2 = points2 + AppConstants.tenPoints;
156
157             numberOfLogs2 = (await firebaseClient
158                 .Child("WaterPoints")
159                     .Child(auth.GetUid())
160                     .OnceSingleAsync<WaterPoints>()).numberOfLogs;
161
162             numberOfLogs2++;
163
164             cisternCount2 = (await firebaseClient
165                 .Child("WaterPoints")
166                     .Child(auth.GetUid())
167                     .OnceSingleAsync<WaterPoints>()).cisternCount;
168
169             rainBarrel2 = (await firebaseClient
170                 .Child("WaterPoints")
171                     .Child(auth.GetUid())
172                     .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
173
174             rainBarrel2++;
175
176             reWater2 = (await firebaseClient
177                 .Child("WaterPoints")
```

```
175     .Child(auth.GetUid())
176     .OnceSingleAsync<WaterPoints>()).reWaterCount;
177 
178     showerBucket2 = (await firebaseClient
179     .Child("WaterPoints")
180     .Child(auth.GetUid())
181     .OnceSingleAsync<WaterPoints>()).showerBucketCount;
182 
183     wSShowerHead2 = (await firebaseClient
184     .Child("WaterPoints")
185     .Child(auth.GetUid())
186     .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
187 
188     await firebaseClient
189     .Child("WaterPoints")
190     .Child(auth.GetUid())
191     .PutAsync(new WaterPoints()
192     {
193         username = username,
194         points = points2,
195         numberOfLogs = numberOfLogs2,
196         cisternCount = cisternCount2,
197         rainBarrelCount = rainBarrel12,
198         reWaterCount = reWater2,
199         showerBucketCount = showerBucket2,
200         wSShowerHeadCount = wSShowerHead2,
201     });
202 }
203 catch (FirebaseException)
204 {
205     username = (await firebaseClient
206     .Child("users")
207     .Child(auth.GetUid())
208     .OnceSingleAsync<Users>()).username;
209 
210     points2 = AppConstants.tenPoints;
211     await firebaseClient
212     .Child("WaterPoints")
213     .Child(auth.GetUid())
214     .PutAsync(new WaterPoints() { username = username, points = points2,
215     numberOfLogs = 1, rainBarrelCount = 1 });
216 }
217 catch (NullReferenceException)
218 {
219     username = (await firebaseClient
220     .Child("users")
221     .Child(auth.GetUid())
222     .OnceSingleAsync<Users>()).username;
223 
224     points2 = AppConstants.tenPoints;
225     await firebaseClient
226     .Child("WaterPoints")
227     .Child(auth.GetUid())
228     .PutAsync(new WaterPoints() { username = username, points = points2,
229     numberOfLogs = 1, rainBarrelCount = 1 });
230 }
231 /**
232  * This function updates the points in the Water category by eight points. It also
233  * increments the number of logs logged in the Water
234  * category by one and increments the number of times this particular action was
235  * logged by one and sends this data to Firebase.
```

```
233     */
234     public async void ReWaterPoints()
235     {
236
237         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
238         quake-default-rtdb.firebaseio.com/");
239         auth = DependencyService.Get<IAuth>();
240
241         try
242         {
243             username = (await firebaseClient
244                 .Child("users")
245                 .Child(auth.GetUid())
246                 .OnceSingleAsync<Users>()).username;
247
248             points2 = (await firebaseClient
249                 .Child("WaterPoints")
250                 .Child(auth.GetUid())
251                 .OnceSingleAsync<WaterPoints>()).points;
252
253             points2 = points2 + AppConstants.eightPoints;
254
255             numberofLogs2 = (await firebaseClient
256                 .Child("WaterPoints")
257                 .Child(auth.GetUid())
258                 .OnceSingleAsync<WaterPoints>()).numberofLogs;
259
260             numberofLogs2++;
261
262             cisternCount2 = (await firebaseClient
263                 .Child("WaterPoints")
264                 .Child(auth.GetUid())
265                 .OnceSingleAsync<WaterPoints>()).cisternCount;
266
267             rainBarrel2 = (await firebaseClient
268                 .Child("WaterPoints")
269                 .Child(auth.GetUid())
270                 .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
271
272             reWater2 = (await firebaseClient
273                 .Child("WaterPoints")
274                 .Child(auth.GetUid())
275                 .OnceSingleAsync<WaterPoints>()).reWaterCount;
276
277             reWater2++;
278
279             showerBucket2 = (await firebaseClient
280                 .Child("WaterPoints")
281                 .Child(auth.GetUid())
282                 .OnceSingleAsync<WaterPoints>()).showerBucketCount;
283
284             wSShowerHead2 = (await firebaseClient
285                 .Child("WaterPoints")
286                 .Child(auth.GetUid())
287                 .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
288
289             await firebaseClient
290                 .Child("WaterPoints")
291                 .Child(auth.GetUid())
292                 .PutAsync(new WaterPoints()
{
```

```
293         username = username,
294         points = points2,
295         numberOfLogs = numberOfLogs2,
296         cisternCount = cisternCount2,
297         rainBarrelCount = rainBarrel2,
298         reWaterCount = reWater2,
299         showerBucketCount = showerBucket2,
300         wSShowerHeadCount = wSShowerHead2,
301     });
302 }
303 catch (FirebaseException)
304 {
305     username = (await firebaseClient
306     .Child("users")
307     .Child(auth.GetUid())
308     .OnceSingleAsync<Users>()).username;
309
310     points2 = AppConstants.eightPoints;
311     await firebaseClient
312     .Child("WaterPoints")
313     .Child(auth.GetUid())
314     .PutAsync(new WaterPoints() { username = username, points = points2,
315     numberOfLogs = 1, reWaterCount = 1 });
316 }
317 catch (NullReferenceException)
318 {
319     username = (await firebaseClient
320     .Child("users")
321     .Child(auth.GetUid())
322     .OnceSingleAsync<Users>()).username;
323
324     points2 = AppConstants.eightPoints;
325     await firebaseClient
326     .Child("WaterPoints")
327     .Child(auth.GetUid())
328     .PutAsync(new WaterPoints() { username = username, points = points2,
329     numberOfLogs = 1, reWaterCount = 1 });
330 }
331 /**
332  * This function updates the points in the Water category by eight points. It also
333  * increments the number of logs logged in the Water
334  * category by one and increments the number of times this particular action was
335  * logged by one and sends this data to Firebase.
336 */
337 public async void ShowerBucketPoints()
338 {
339
340     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
341     quake-default-rtdb.firebaseio.com/");
342     auth = DependencyService.Get<IAuth>();
343
344     try
345     {
346         username = (await firebaseClient
347         .Child("users")
348         .Child(auth.GetUid())
349         .OnceSingleAsync<Users>()).username;
350
351         points2 = (await firebaseClient
352         .Child("WaterPoints")
353         .Child(auth.GetUid())
354         .PutAsync(new WaterPoints() { username = username, points = points2,
355         numberOfLogs = 1, reWaterCount = 1 }));
356     }
357     catch (NullReferenceException)
358     {
359         username = (await firebaseClient
360         .Child("users")
361         .Child(auth.GetUid())
362         .OnceSingleAsync<Users>()).username;
363
364         points2 = (await firebaseClient
365         .Child("WaterPoints")
366         .Child(auth.GetUid())
367         .PutAsync(new WaterPoints() { username = username, points = points2,
368         numberOfLogs = 1, reWaterCount = 1 }));
369     }
370 }
```

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

```
411     await firebaseClient
412         .Child("WaterPoints")
413         .Child(auth.GetUid())
414         .PutAsync(new WaterPoints() { username = username, points = points2,
415     numberOfLogs = 1, showerBucketCount = 1 });
416 }
417 catch (NullReferenceException)
418 {
419     username = (await firebaseClient
420         .Child("users")
421         .Child(auth.GetUid())
422         .OnceSingleAsync<Users>()).username;
423
424     points2 = AppConstants.eightPoints;
425     await firebaseClient
426         .Child("WaterPoints")
427         .Child(auth.GetUid())
428         .PutAsync(new WaterPoints() { username = username, points = points2,
429     numberOfLogs = 1, showerBucketCount = 1 });
430 }
431 /**
432 * This function updates the points in the Water category by ten points. It also
433 increments the number of logs logged in the Water
434 * category by one and increments the number of times this particular action was
435 logged by one and sends this data to Firebase.
436 */
437 public async void WSSowerHeadPoints()
438 {
439
440     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
441     quake-default-rtdb.firebaseio.com/");
442     auth = DependencyService.Get<IAuth>();
443
444     try
445     {
446         username = (await firebaseClient
447             .Child("users")
448             .Child(auth.GetUid())
449             .OnceSingleAsync<Users>()).username;
450
451         points2 = (await firebaseClient
452             .Child("WaterPoints")
453             .Child(auth.GetUid())
454             .OnceSingleAsync<WaterPoints>()).points;
455
456         points2 = points2 + AppConstants.tenPoints;
457
458         numberOfLogs2 = (await firebaseClient
459             .Child("WaterPoints")
460             .Child(auth.GetUid())
461             .OnceSingleAsync<WaterPoints>()).numberOfLogs;
462
463         numberOfLogs2++;
464
465         cisternCount2 = (await firebaseClient
466             .Child("WaterPoints")
467             .Child(auth.GetUid())
468             .OnceSingleAsync<WaterPoints>()).cisternCount;
469
470         rainBarrel2 = (await firebaseClient
471             .Child("WaterPoints")
```

```
468     .Child(auth.GetUid())
469     .OnceSingleAsync<WaterPoints>()).rainBarrelCount;
470 
471     reWater2 = (await firebaseClient
472     .Child("WaterPoints")
473     .Child(auth.GetUid())
474     .OnceSingleAsync<WaterPoints>()).reWaterCount;
475 
476     showerBucket2 = (await firebaseClient
477     .Child("WaterPoints")
478     .Child(auth.GetUid())
479     .OnceSingleAsync<WaterPoints>()).showerBucketCount;
480 
481     wSShowerHead2 = (await firebaseClient
482     .Child("WaterPoints")
483     .Child(auth.GetUid())
484     .OnceSingleAsync<WaterPoints>()).wSShowerHeadCount;
485 
486     wSShowerHead2++;
487 
488     await firebaseClient
489     .Child("WaterPoints")
490     .Child(auth.GetUid())
491     .PutAsync(new WaterPoints()
492     {
493         username = username,
494         points = points2,
495         numberLogs = numberLogs2,
496         cisternCount = cisternCount2,
497         rainBarrelCount = rainBarrel2,
498         reWaterCount = reWater2,
499         showerBucketCount = showerBucket2,
500         wSShowerHeadCount = wSShowerHead2,
501     });
502 }
503 catch (FirebaseException)
504 {
505     username = (await firebaseClient
506     .Child("users")
507     .Child(auth.GetUid())
508     .OnceSingleAsync<Users>()).username;
509 
510     points2 = AppConstants.tenPoints;
511     await firebaseClient
512     .Child("WaterPoints")
513     .Child(auth.GetUid())
514     .PutAsync(new WaterPoints() { username = username, points = points2,
515     numberLogs = 1, wSShowerHeadCount = 1 });
516 }
517 catch (NullReferenceException)
518 {
519     username = (await firebaseClient
520     .Child("users")
521     .Child(auth.GetUid())
522     .OnceSingleAsync<Users>()).username;
523 
524     points2 = AppConstants.tenPoints;
525     await firebaseClient
526     .Child("WaterPoints")
527     .Child(auth.GetUid())
```

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

```
1 /*! \class The WorkPointsUpdate ViewModel Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the WorkPointsUpdate ViewModel Class. It updates the data for the
8  * Work Category of the application. The functions in this class
9  * work by reading in all the chosen data and updating the selected fields and then sending
10 * this data to back firebase.
11 */
12
13 using Application_Green_Quake.Models;
14 using Firebase.Database;
15 using Firebase.Database.Query;
16 using System;
17 using Xamarin.Forms;
18
19 namespace Application_Green_Quake.ViewModels
20 {
21     class WorkPointsUpdate
22     {
23         int points2 = 0;
24         int numberOfWorkLogs2 = 0;
25         int paperCount2 = 0;
26         int offElectronicsCount2 = 0;
27         int remoteWorkCount2 = 0;
28
29         string username = "";
30
31         IAuth auth;
32         /** This function updates the points in the Work category by four points. It also
33          increments the number of logs logged in the Work
34          * category by one and increments the number of times this particular action was
35          logged by one and sends this data to Firebase.
36         */
37         public async void PaperPoints()
38         {
39
40             FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
41             quake-default-rtdb.firebaseio.com/");
42             auth = DependencyService.Get<IAuth>();
43
44             try
45             {
46                 username = (await firebaseClient
47                     .Child("users")
48                     .Child(auth.GetUid())
49                     .OnceSingleAsync<Users>()).username;
50
51                 points2 = (await firebaseClient
52                     .Child("WorkPoints")
53                     .Child(auth.GetUid())
54                     .OnceSingleAsync<WorkPoints>()).points;
55
56                 points2 = points2 + AppConstants.fourPoints;
57
58                 numberOfWorkLogs2 = (await firebaseClient
59                     .Child("WorkPoints")
60                     .Child(auth.GetUid())
61                     .OnceSingleAsync<WorkPoints>()).numberOfLogs;
```

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

```
117     /** This function updates the points in the Work category by six points. It also  
118      increments the number of logs logged in the Work  
119      * category by one and increments the number of times this particular action was  
120      logged by one and sends this data to Firebase.  
121      */  
122      public async void ElectronicsOffPoints()  
123      {  
124  
125          FirebaseClient firebaseClient = new FirebaseClient("https://application-green-  
126          quake-default-rtdb.firebaseio.com/");  
127          auth = DependencyService.Get<IAuth>();  
128  
129          try  
130          {  
131              username = (await firebaseClient  
132                  .Child("users")  
133                  .Child(auth.GetUid())  
134                  .OnceSingleAsync<Users>()).username;  
135  
136              points2 = (await firebaseClient  
137                  .Child("WorkPoints")  
138                  .Child(auth.GetUid())  
139                  .OnceSingleAsync<WorkPoints>()).points;  
140  
141              points2 = points2 + AppConstants.sixPoints;  
142  
143              numberOfWorkLogs2 = (await firebaseClient  
144                  .Child("WorkPoints")  
145                  .Child(auth.GetUid())  
146                  .OnceSingleAsync<WorkPoints>()).numberOfWorkLogs;  
147  
148              numberOfWorkLogs2++;  
149  
150              paperCount2 = (await firebaseClient  
151                  .Child("WorkPoints")  
152                  .Child(auth.GetUid())  
153                  .OnceSingleAsync<WorkPoints>()).paperCount;  
154  
155              offElectronicsCount2 = (await firebaseClient  
156                  .Child("WorkPoints")  
157                  .Child(auth.GetUid())  
158                  .OnceSingleAsync<WorkPoints>()).offElectronicsCount;  
159  
160              offElectronicsCount2++;  
161  
162              remoteWorkCount2 = (await firebaseClient  
163                  .Child("WorkPoints")  
164                  .Child(auth.GetUid())  
165                  .OnceSingleAsync<WorkPoints>()).remoteWorkCount;  
166  
167              await firebaseClient  
168                  .Child("WorkPoints")  
169                  .Child(auth.GetUid())  
170                  .PutAsync(new WorkPoints()  
171                  {  
172                      username = username,  
173                      points = points2,  
174                      numberOfWorkLogs = numberOfWorkLogs2,  
175                      paperCount = paperCount2,  
176                      offElectronicsCount = offElectronicsCount2,  
177                      remoteWorkCount = remoteWorkCount,  
178                  });  
179      }
```

```
176     }
177     catch (FirebaseException)
178     {
179         username = (await firebaseClient
180             .Child("users")
181             .Child(auth.GetUid())
182             .OnceSingleAsync<Users>()).username;
183
184         points2 = AppConstants.sixPoints;
185         await firebaseClient
186             .Child("WorkPoints")
187             .Child(auth.GetUid())
188             .PutAsync(new WorkPoints() { username = username, points = points2,
numberOfLogs = 1, offElectronicsCount = 1 });
189     }
190     catch (NullReferenceException)
191     {
192         username = (await firebaseClient
193             .Child("users")
194             .Child(auth.GetUid())
195             .OnceSingleAsync<Users>()).username;
196
197         points2 = AppConstants.sixPoints;
198         await firebaseClient
199             .Child("WorkPoints")
200             .Child(auth.GetUid())
201             .PutAsync(new WorkPoints() { username = username, points = points2,
numberOfLogs = 1, offElectronicsCount = 1 });
202     }
203     */
204     /**
205      * This function updates the points in the Work category by ten points. It also
206      * increments the number of logs logged in the Work
207      * category by one and increments the number of times this particular action was
208      * logged by one and sends this data to Firebase.
209      */
210     public async void RemoteWorkPoints()
211     {
212
213         FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
214         quake-default-rtdb.firebaseio.com/");
215         auth = DependencyService.Get<IAuth>();
216
217         try
218         {
219             username = (await firebaseClient
220                 .Child("users")
221                 .Child(auth.GetUid())
222                 .OnceSingleAsync<Users>()).username;
223
224             points2 = (await firebaseClient
225                 .Child("WorkPoints")
226                 .Child(auth.GetUid())
227                 .OnceSingleAsync<WorkPoints>()).points;
228
229             points2 = points2 + AppConstants.tenPoints;
230
231             numberOfLogs2 = (await firebaseClient
232                 .Child("WorkPoints")
233                 .Child(auth.GetUid())
234                 .OnceSingleAsync<WorkPoints>()).numberOfLogs;
235         }
236     }
```

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

```
293 }  
294 }
```

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

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

```

1 /*! \class The SignUpPage View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * \date 28/04/2021
4  * \section desc_sec Description
5  *
6  * Description: This is the SignUpPage View Class. This is the class that allows a user to
7  * sign up for the application. It contains validation checks.
8  */
9 using Application_Green_Quake.Models;
10 using Firebase.Database;
11 using Firebase.Database.Query;
12 using System;
13 using System.Globalization;
14 using System.Text.RegularExpressions;
15 using Acr.UserDialogs;
16 using Xamarin.Forms;
17 using Xamarin.Forms.Xaml;
18
19 namespace Application_Green_Quake.Views
20 {
21     [XamlCompilation(XamlCompilationOptions.Compile)]
22     public partial class SignUpPage : ContentPage
23     {
24         IAuth auth;
25         public SignUpPage()
26         {
27             InitializeComponent();
28             auth = DependencyService.Get<IAuth>();
29         }
30         /** This function fires when the Sign Up button is clicked. It carries out
validation checks and if all are passed the the new user is created. If not
* the correct error message is displayed.
31         */
32         async void SignUpClicked(object sender, EventArgs e)
33         {
34             var emailPattern = "^(?(\")(\\".+?(?<!\\\\\\\")\"@)|(([0-9a-z]((\\.\\.(?!\\.)\\.)|[-!#\\$%&'\\\\*\\\\+/=\\\\?\\\\^\\\\{\\\\}\\\\|~\\\\w])*)|(\\?<=[0-9a-z])@))|(\\?([\\\\](\\\\[\\\\d{1,3}\\\\.]\\.)\\{3}\\d{1,3}\\\\]))|(([0-9a-z][-\\\\w]*[0-9a-z]*\\\\.)(+[a-z0-9][\\\\-a-z0-9]\\{0,22\\}[a-z0-9]))$";
35             var hasNum = new Regex(@"[0-9]+");
36             var hasUpperChar = new Regex(@"[A-Z]+");
37             var hasLowerChar = new Regex(@"[a-z]+");
38             var hasSpecialChar = new Regex(@"[^\\w]+");
39             var hasMinimum8Chars = new Regex(@".{8,}");
40
41             UsernameErrorLabel.Text = null;
42             EmailErrorLabel.Text = null;
43             PasswordErrorLabel.Text = null;
44
45             if (UsernameInput.Text == null && EmailInput.Text != null && PasswordInput.Text
!= null)
46             {
47                 UsernameInput.Text = null;
48                 UsernameErrorLabel.Text = "No Username Entered";
49             }
50             else if (UsernameInput.Text != null && EmailInput.Text == null &&
PasswordInput.Text != null)
51             {
52                 EmailInput.Text = null;
53                 EmailErrorLabel.Text = "No Email Entered";
54             }
55         }

```

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

```

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

```

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

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

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

```
1 /*! \class The MainPage View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the MainPage View Class. This is the class that allows a user to
8  * login to the application. It contains validation checks.
9 */
10 using System;
11 using Application_Green_Quake.ViewModels;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class MainPage : ContentPage
19     {
20         IAuth auth;
21         public static string token;
22         public MainPage()
23         {
24             InitializeComponent();
25             auth = DependencyService.Get<IAuth>();
26         }
27         /** This function fires when the Login button is clicked. It carries out validation
28         checks and if all are passed the the user is logged in. If not
29             * the correct error message is displayed.
30         */
31         async void LoginClicked(object sender, EventArgs e)
32         {
33             EmailErrorLabel.Text = null;
34             PasswordErrorLabel.Text = null;
35
36             if (EmailInput.Text == null && PasswordInput.Text != null)
37             {
38                 EmailInput.Text = null;
39                 PasswordInput.Text = null;
40                 EmailErrorLabel.Text = "No Email Entered";
41             }
42             else if (PasswordInput.Text == null && EmailInput.Text != null)
43             {
44                 EmailInput.Text = null;
45                 PasswordInput.Text = null;
46                 PasswordErrorLabel.Text = "No Password Entered";
47             }
48             else if (EmailInput.Text == null && PasswordInput.Text == null)
49             {
50                 EmailInput.Text = null;
51                 PasswordInput.Text = null;
52                 EmailErrorLabel.Text = "No Email Entered";
53                 PasswordErrorLabel.Text = "No Password Entered";
54             }
55             else
56             {
57                 try
58                 {
59                     token = await auth.LoginWithEmailAndPassword(EmailInput.Text,
60                     PasswordInput.Text);
61                 }
62             }
63         }
64     }
65 }
```

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

```
1 <?xml version="1.0" encoding="utf-8" ?>
2
3 <TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
4         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:views="clr-
5 namespace:Application_Green_Quake.Views"
6         xmlns:local="clr-namespace:Application_Green_Quake.Models" xmlns:views1="clr-
7 namespace:Application_Green_Quake.Views.ProfilePage" xmlns:views2="clr-
8 namespace:Application_Green_Quake.Views.LeaderboardPage"
9             x:Class="Application_Green_Quake.Views.MainMenu"
10            NavigationPage.HasBackButton="False"
11            xmlns:android="clr-
12 namespace:Xamarin.Forms.PlatformConfiguration.AndroidSpecific;assembly=Xamarin.Forms.Core"
13             android:TabbedPage.ToolbarPlacement="Bottom"
14             UnselectedTabColor="Black"
15             SelectedTabColor="White" >
16
17     <NavigationPage.TitleView>
18         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
19             VerticalOptions="EndAndExpand" Spacing="0">
20             <Label Text="Green Quake" TextColor="White" FontSize="20"
21                 FontAttributes="Italic" VerticalOptions="CenterAndExpand"
22                 HorizontalOptions="StartAndExpand"/>
23             <Label x:Name="theLevel" TextColor="White" FontSize="20"
24                 FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
25                 Margin="0,0,30,0"/>
26         </StackLayout>
27     </NavigationPage.TitleView>
28
29     <ContentPage Title="Home" IconImageSource="{local:ImageResource
30 Application_Green_Quake.Images.Tabbed.Home.png}">
31         <ScrollView>
32             <Grid VerticalOptions="FillAndExpand" RowSpacing="5">
33                 <Grid.RowDefinitions>
34                     <RowDefinition Height="2*"/>
35                     <RowDefinition Height="2*"/>
36                     <RowDefinition Height="*"/>
37                 </Grid.RowDefinitions>
38
39                 <Image Grid.Column="0"
40                     Grid.Row="0"
41                     Aspect="AspectFill"
42                     Source="{local:ImageResource
43 Application_Green_Quake.Images.EcoActions.jpg}"/>
44                 <StackLayout Grid.Column="0"
45                     Grid.Row="0"
46                     BackgroundColor="Black"
47                     Opacity=".4">
48                     <StackLayout.GestureRecognizers>
49                         <TapGestureRecognizer Tapped="NavigateToEcoActionButton"/>
50                     </StackLayout.GestureRecognizers>
51                 </StackLayout>
52                 <StackLayout Grid.Column="0"
53                     Grid.Row="0"
54                     VerticalOptions="Center"
55                     Spacing="5"
56                     Margin="15,0,40,15">
57                     <StackLayout.GestureRecognizers>
58                         <TapGestureRecognizer Tapped="NavigateToEcoActionButton"/>
59                     </StackLayout.GestureRecognizers>
60                     <Label Text="Eco Actions"
61                         TextColor="White"
62                         FontSize="24"
```

```

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

```

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

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

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

```
1 /*! \class The LoginSplashPage View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the LoginSplashPage View Class. This class provides the splash
8  * screen when a user logs into the application.
9  */
10
11 using Xamarin.Forms;
12 {
13     public class LoginSplashPage : ContentPage
14     {
15         Image splashImage;
16
17         /** This function fires when the Login button is clicked. It provides the splash
18         screen and then navigates to the main menu right after.
19         */
20         public LoginSplashPage()
21         {
22             NavigationPage.SetHasNavigationBar(this, false);
23
24             //Set the image for the splash screen
25             var sub = new AbsoluteLayout();
26             splashImage = new Image
27             {
28                 Source =
29                 ImageSource.FromResource("Application_Green_Quake.Images.trees.png"),
30                 WidthRequest = 150,
31                 HeightRequest = 150
32             };
33
34             AbsoluteLayout.SetLayoutFlags(splashImage,
35                 AbsoluteLayoutFlags.PositionProportional);
36             AbsoluteLayout.SetLayoutBounds(splashImage,
37                 new Rectangle(0.5, 0.5, AbsoluteLayout.AutoSize, AbsoluteLayout.AutoSize));
38
39             sub.Children.Add(splashImage);
40
41             this.BackgroundColor = Color.FromHex("#50C878");
42             this.Content = sub;
43         }
44
45         protected override async void OnAppearing()
46         {
47             base.OnAppearing();
48             //Set the animation
49             await splashImage.ScaleTo(0.4, 1100, Easing.Linear);
50             await splashImage.ScaleTo(700, 900, Easing.Linear);
51             Application.Current.MainPage = new NavigationPage(new MainMenu());
52         }
53     }
54 }
```

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

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

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

```

1 /*! \class The RefillStation View Class
2 * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
c00228956@itcarlow.ie
3 * \date 28/04/2021
4 * \section desc_sec Description
5 *
6 * Description: This is the RefillStation View Class. This page contains the map and loads
data from firebase and then uses this data to display the pins.
7 *
8 */
9 using Application_Green_Quake.Models;
10 using Firebase.Database;
11 using System;
12 using System.Diagnostics;
13 using System.Linq;
14 using Xamarin.Essentials;
15 using Xamarin.Forms;
16 using Xamarin.Forms.GoogleMaps;
17 using Xamarin.Forms.Xaml;
18 namespace Application_Green_Quake.Views.RefillPage
19 {
20     [XamlCompilation(XamlCompilationOptions.Compile)]
21     public partial class RefillStation : ContentPage
22     {
23         public RefillStation()
24         {
25             InitializeComponent();
26             OnAppearing();
27         }
28         /** This function is called before the page is displayed.
29         */
30         protected override async void OnAppearing()
31         {
32             try
33             {
34                 var location = await Geolocation.GetLastKnownLocationAsync();
35                 if (location == null)
36                 {
37                     location = await Geolocation.GetLocationAsync(new GeolocationRequest
38                     {
39                         DesiredAccuracy = GeolocationAccuracy.Medium,
40                         Timeout = TimeSpan.FromSeconds(30)
41                     });
42
43
44                 }
45                 else
46                 {
47                     Pin currentLocation = new Pin()
48                     {
49                         Type = PinType.SavedPin,
50                         Label = "Me",
51                         Address = "Here",
52                         Position = new Position(location.Latitude, location.Longitude),
53                         Tag = "id_Me",
54
55                     };
56                     // Add the pin and load the map at this location
57                     map.Pins.Add(currentLocation);
58                     map.MoveToRegion(MapSpan.FromCenterAndRadius(currentLocation.Position,
Distance.FromMeters(5000)));
59                 }
60             }
61         }
62     }
63 }
```

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

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

```
1 /*! \class The AirOutHome View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the AirOutHome View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class AirOutHome : ContentPage
20     {
21         public AirOutHome()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTwoPoints();
55                 HomePointsUpdate helper2 = new HomePointsUpdate();
56                 helper2.AirOutPoints();
57                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The BillsOnline View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the BillsOnline View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class BillsOnline : ContentPage
20     {
21         public BillsOnline()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 WastePointsUpdate helper2 = new WastePointsUpdate();
56                 helper2.BillsPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

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

4/29/2021

c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Work\BothSidesPaper.xaml.cs

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

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

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

```
58     }
59     /** This function is called before the page is displayed and it creates an object
60     ans uses its SetLvl method to set the players level in the app
61     * and display it in the navigation bar.
62     */
63     protected override void OnAppearing()
64     {
65         GetData data = new GetData();
66         data.SetLvl();
67
68         theLevel.Text = "LVL: " + GetData.lvl.ToString();
69     }
70 }
```

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

```
1 /*! \class The BuyOrganicFood View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the BuyOrganicFood View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class BuyOrganicFood : ContentPage
20     {
21         public BuyOrganicFood()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.OrganicPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\BuyOrganicFood....  
58     /** This function is called before the page is displayed and it creates an object  
59     ans uses its SetLvl method to set the players level in the app  
60     * and display it in the navigation bar.  
61     */  
62     protected override void OnAppearing()  
63     {  
64         GetData data = new GetData();  
65         data.SetLvl();  
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();  
67     }  
68 }  
69 }
```

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

```
1 /*! \class The Carpool View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the Carpool View Class. This class is the eco action that the user
8  * can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Travel
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class Carpool : ContentPage
20     {
21         public Carpool()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 TravelPointsUpdate helper2 = new TravelPointsUpdate();
56                 helper2.CarpoolPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The CisternDisplacement View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the CisternDisplacement View Class. This class is the eco action
8  * that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class CisternDisplacement : ContentPage
20     {
21         public CisternDisplacement()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
56                 helper2.CisternPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

4/29/2021 c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Water\CisternDisplacement.xaml.cs

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

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

```
1 /*! \class The ClothNapkins View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ClothNapkins View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ClothNapkins : ContentPage
20     {
21         public ClothNapkins()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTwoPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.ClothNapkinsPoints();
57                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The ClothTowels View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ClothTowels View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ClothTowels : ContentPage
20     {
21         public ClothTowels()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTwoPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.ClothTowelsPoints();
57                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The CompostWaste View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the CompostWaste View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class CompostWaste : ContentPage
20     {
21         public CompostWaste()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 WastePointsUpdate helper2 = new WastePointsUpdate();
56                 helper2.CompostPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The CreateEnvironmentalGroup View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the CreateEnvironmentalGroup View Class. This class is the eco
8  * action that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class CreateEnvironmentalGroup : ContentPage
20     {
21         public CreateEnvironmentalGroup()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
56                 helper2.CreateGroupPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Community\CreateEnvironmental...  
58     /** This function is called before the page is displayed and it creates an object  
59     ans uses its SetLvl method to set the players level in the app  
60     * and display it in the navigation bar.  
61     */  
62     protected override void OnAppearing()  
63     {  
64         GetData data = new GetData();  
65         data.SetLvl();  
66         theLevel.Text = "LVL: " + GetData.lvl.ToString();  
67     }  
68 }  
69 }
```

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

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

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

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

```
1 /*! \class The DishwasherFull View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the DishwasherFull View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Habits
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class DishwasherFull : ContentPage
20     {
21         public DishwasherFull()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
56                 helper2.DishWasherFullPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Habits\DishwasherFull.xaml.cs
58  /** This function is called before the page is displayed and it creates an object
59  ans uses its SetLvl method to set the players level in the app
60  * and display it in the navigation bar.
61  */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The DoCommunity View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the DoCommunity View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class DoCommunity : ContentPage
20     {
21         public DoCommunity()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
56                 helper2.CommunityPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Community\DoCommunity.xaml.cs
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The DonateItems View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the DonateItems View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class DonateItems : ContentPage
20     {
21         public DonateItems()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
56                 helper2.DonatePoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Community\DonateItems.xaml.cs
58  /* This function is called before the page is displayed and it creates an object
59  ans uses its SetLvl method to set the players level in the app
60  * and display it in the navigation bar.
61  */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The DryerFull View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the DryerFull View Class. This class is the eco action that the user
8  * can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class DryerFull : ContentPage
20     {
21         public DryerFull()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.DryerFullPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4         x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.EatAllYouMake"
5         xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10            <Label Text="Eat All" TextColor="White" FontSize="20" FontAttributes="Italic"
11            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13            VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14        </StackLayout>
15    </NavigationPage.TitleView>
16
17    <ContentPage.Content>
18        <ScrollView>
19            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                <Grid.RowDefinitions>
21                    <RowDefinition Height="2*"/>
22                    <RowDefinition Height="2*"/>
23                    <RowDefinition Height="*"/>
24                </Grid.RowDefinitions>
25
26                <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.FD.EatAllYouMake.jpg}">
29                    <StackLayout Grid.Row="1"
30                         VerticalOptions="Center"
31                         HorizontalOptions="Center"
32                         Margin="15,0,15,0"
33                         Padding="0,5,0,5"
34                         BackgroundColor="#D3D3D3">
35                        <Label Text="Only make enough food so you can eat it all. This reduces
36 food waste which is a major concern. When food rots it emits methane, a powerful greenhouse
37 gas which is 25 times worse for the ozone layer than CO2. 60% of methane is produced by
38 humans.">
39                        TextColor="Black"
40                        HorizontalTextAlignment="Center"
41                        FontSize="20"/>
42                    </StackLayout>
43                    <StackLayout Grid.Row="2"
44                         Spacing="10">
45                        <Label Text="4 POINTS!">
46                            TextColor="Black"
47                            FontAttributes="Bold"
48                            FontSize="20"
49                            HorizontalOptions="Center"/>
50                        <Button Text="Completed"
51                             BackgroundColor="#50C878"
52                             TextColor="White"
53                             Margin="60,0,60,0"
54                             Clicked="AddPointsClicked"/>
55                    </StackLayout>
56                </Grid>
57            </ScrollView>
58        </ContentPage.Content>
59    </ContentPage>
```

```
1 /*! \class The EatAllYouMake View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the EatAllYouMake View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EatAllYouMake : ContentPage
20     {
21         public EatAllYouMake()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.EatAllPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\EatAllYouMake.xaml.cs
58  /// ** This function is called before the page is displayed and it creates an object
59  /// and uses its SetLvl method to set the player's level in the app
60  /// * and display it in the navigation bar.
61  /// 
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66      theLevel.Text = "LVL: " + GetData.lvl.ToString();
67  }
68 }
69 }
```

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

```
1 /*! \class The EcoFreidnlyApplicance View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the EcoFreidnlyApplicance View Class. This class is the eco action
8  * that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EcoFreidnlyApplicance : ContentPage
20     {
21         public EcoFreidnlyApplicance()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.AppliancePoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Shopping\EcoFreidnlyApplicance...
58  /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59   * methods
60   */
61  protected override void OnAppearing()
62  {
63      GetData data = new GetData();
64      data.SetLvl();
65
66      theLevel.Text = "LVL: " + GetData.lvl.ToString();
67  }
68 }
69 }
```

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

```
1 /*! \class The EcoFreindlyCar View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the EcoFreindlyCar View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Travel
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EcoFreindlyCar : ContentPage
20     {
21         public EcoFreindlyCar()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 TravelPointsUpdate helper2 = new TravelPointsUpdate();
56                 helper2.EcoCarPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The EcoFriendlyProduct View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the EcoFriendlyProduct View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EcoFriendlyProduct : ContentPage
20     {
21         public EcoFriendlyProduct()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.ProductPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Shopping\EcoFriendlyProduct.xa...
58  /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59   * methods
60   */
61  protected override void OnAppearing()
62  {
63      GetData data = new GetData();
64      data.SetLvl();
65
66      theLevel.Text = "LVL: " + GetData.lvl.ToString();
67  }
68 }
69 }
```

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

```
1 /*! \class The EcoFriendlyToothbrush View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the EcoFriendlyToothbrush View Class. This class is the eco action
8  * that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EcoFriendlyToothbrush : ContentPage
20     {
21         public EcoFriendlyToothbrush()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.EcoToothbrushPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Shopping\EcoFriendlyToothbrush...
58  /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59   * methods
60   */
61  protected override void OnAppearing()
62  {
63      GetData data = new GetData();
64      data.SetLvl();
65
66      theLevel.Text = "LVL: " + GetData.lvl.ToString();
67  }
68 }
69 }
```

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

```
1 /*! \class The EfficientThermostat View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the EfficientThermostat View Class. This class is the eco action
8  * that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EfficientThermostat : ContentPage
20     {
21         public EfficientThermostat()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.EfficientThermostatPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Energy\EfficientThermostat.xaml.cs
58  ///** This function is called before the page is displayed and it creates an object
59  //ans uses its SetLvl method to set the players level in the app
60  //    * and display it in the navigation bar.
61  //*/
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

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

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Community\EnvironmentalGroup...
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4         x:Class="Application_Green_Quake.Views.EcoActions.Shopping.EthicalClothes"
5         xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10            <Label Text="Ethical Clothes" TextColor="White" FontSize="20"
11                FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12                HorizontalOptions="StartAndExpand"/>
13                <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14                    VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15            </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*"/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28                     Source="{local:ImageResource
29                         Application_Green_Quake.Images.SubCategories.Shopping.EthicalClothes.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                     VerticalOptions="Center"
32                     HorizontalOptions="Center"
33                     Margin="15,0,15,0"
34                     Padding="0,5,0,5"
35                     BackgroundColor="#D3D3D3">
36                     <Label Text="Purchase and wear ethical clothes. Ethical Clothing is an
37                         umbrella term to describe ethical fashion design, production, retail, and purchasing. It
38                         covers a range of issues such as working conditions, exploitation, fair trade, sustainable
39                         production, the environment, and animal welfare."
40                     TextColor="Black"
41                     HorizontalTextAlignment="Center"
42                     FontSize="20"/>
43                 </StackLayout>
44                 <StackLayout Grid.Row="2"
45                     Spacing="10">
46                     <Label Text="10 POINTS!">
47                         TextColor="Black"
48                         FontAttributes="Bold"
49                         FontSize="20"
50                         HorizontalOptions="Center"/>
51                     <Button Text="Completed"
52                         BackgroundColor="#50C878"
53                         TextColor="White"
54                         Margin="60,0,60,0"
55                         Clicked="AddPointsClicked"/>
56                 </StackLayout>
57             </Grid>
58         </ScrollView>
59     </ContentPage.Content>
60 </ContentPage>
```

```
1 /*! \class The EthicalClothes View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the EthicalClothes View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class EthicalClothes : ContentPage
20     {
21         public EthicalClothes()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.ClothesPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Shopping\EthicalClothes.xaml.cs
58  /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
59   * methods
60   */
61  protected override void OnAppearing()
62  {
63      GetData data = new GetData();
64      data.SetLvl();
65
66      theLevel.Text = "LVL: " + GetData.lvl.ToString();
67  }
68 }
69 }
```

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

```
1 /*! \class The FixInsteadOfThrowAway View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the FixInsteadOfThrowAway View Class. This class is the eco action
8  * that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.AdvancedPageItems
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class FixInsteadOfThrowAway : ContentPage
20     {
21         public FixInsteadOfThrowAway()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 AdvancedPointsUpdate helper2 = new AdvancedPointsUpdate();
56                 helper2.FixPoints();
57                 GetData data = new GetData();
58                 data.SetLvl();
59                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
60                 await Navigation.PushAsync(new MainMenu());
61             }
62         }
63     }
64 }
```

```
58         }
59     }
60     /** This function is called before the page is displayed and it creates an object
61     ans uses its SetLvl method to set the players level in the app
62     * and display it in the navigation bar.
63     */
64     protected override void OnAppearing()
65     {
66         GetData data = new GetData();
67         data.SetLvl();
68
69         theLevel.Text = "LVL: " + GetData.lvl.ToString();
70     }
71 }
```

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

```
1 /*! \class The FoodDelivered View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the FoodDelivered View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class FoodDelivered : ContentPage
20     {
21         public FoodDelivered()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.FoodDelivredPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\FoodDelivered.xa...
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

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

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

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

```
1 /*! \class The GoCamping View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the GoCamping View Class. This class is the eco action that the user
8  * can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class GoCamping : ContentPage
20     {
21         public GoCamping()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.CampingPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Outdoors\GoCamping.xaml.cs
58  /** This function is called before the page is displayed and it creates an object
59  ans uses its SetLvl method to set the players level in the app
60  * and display it in the navigation bar.
61  */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The HangDry View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the HangDry View Class. This class is the eco action that the user
8  * can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class HangDry : ContentPage
20     {
21         public HangDry()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.HangDryPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The InsulateWater View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the InsulateWater View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class InsulateWater : ContentPage
20     {
21         public InsulateWater()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.InsulateWaterPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The IsolateHome View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the IsolateHome View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class IsolateHome : ContentPage
20     {
21         public IsolateHome()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.IsolateHomePoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The LedLightBulb View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the LedLightBulb View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class LedLightBulb : ContentPage
20     {
21         public LedLightBulb()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.LedLightsPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The LocalProduct View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the LocalProduct View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class LocalProduct : ContentPage
20     {
21         public LocalProduct()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.LocalProductPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The LooseLeafTea View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the LooseLeafTea View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class LooseLeafTea : ContentPage
20     {
21         public LooseLeafTea()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.TeaPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The MachineFull View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the MachineFull View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class MachineFull : ContentPage
20     {
21         public MachineFull()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.MachineFullPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The MicrowaveNotOven View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the MicrowaveNotOven View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class MicrowaveNotOven : ContentPage
20     {
21         public MicrowaveNotOven()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.MicrowavePoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Energy\MicrowaveNotOven.xaml.cs
58  /* This function is called before the page is displayed and it creates an object
59  ans uses its SetLvl method to set the players level in the app
60  * and display it in the navigation bar.
61  */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The NoMeat View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the NoMeat View Class. This class is the eco action that the user
8  * can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class NoMeat : ContentPage
20     {
21         public NoMeat()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.NoMeatPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\NoMeat.xaml.cs
58  ///** This function is called before the page is displayed and it creates an object
59  //ans uses its SetLvl method to set the players level in the app
60  //    * and display it in the navigation bar.
61  //*/
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The NonHarmfulProducts View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the NonHarmfulProducts View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class NonHarmfulProducts : ContentPage
20     {
21         public NonHarmfulProducts()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 HomePointsUpdate helper2 = new HomePointsUpdate();
56                 helper2.NonHarmfulPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Home\NonHarmfulProducts.xaml.cs
58  ///** This function is called before the page is displayed and it creates an object
59  //ans uses its SetLvl method to set the players level in the app
60  //    * and display it in the navigation bar.
61  //*/
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

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

4/29/2021

c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Work\OffElectronics.xaml.cs

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

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4         x:Class="Application_Green_Quake.Views.EcoActions.Energy.OffSocketSwitch"
5         xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10            <Label Text="Socket Off" TextColor="White" FontSize="20" FontAttributes="Italic"
11                VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13                VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14        </StackLayout>
15    </NavigationPage.TitleView>
16
17    <ContentPage.Content>
18        <ScrollView>
19            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                <Grid.RowDefinitions>
21                    <RowDefinition Height="2*"/>
22                    <RowDefinition Height="2*"/>
23                    <RowDefinition Height="*"/>
24                </Grid.RowDefinitions>
25
26                <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28                         Application_Green_Quake.Images.SubCategories.Energy.OffSocket.jpg}"/>
29
30                <StackLayout Grid.Row="1"
31                            VerticalOptions="Center"
32                            HorizontalOptions="Center"
33                            Margin="15,0,15,0"
34                            Padding="0,5,0,5"
35                            BackgroundColor="#D3D3D3">
36                    <Label Text="Turn off the wall sockets that are not in use or plug out
37 the device if the socket cannot be switched off. This will save energy from the devices
38 stand by mode."
39                        TextColor="Black"
40                        HorizontalTextAlignment="Center"
41                        FontSize="20"/>
42                </StackLayout>
43                <StackLayout Grid.Row="2"
44                            Spacing="10">
45                    <Label Text="4 POINTS!">
46                        TextColor="Black"
47                        FontAttributes="Bold"
48                        FontSize="20"
49                        HorizontalOptions="Center"/>
50                    <Button Text="Completed"
51                            BackgroundColor="#50C878"
52                            TextColor="White"
53                            Margin="60,0,60,0"
54                            Clicked="AddPointsClicked"/>
55                </StackLayout>
56            </Grid>
57        </ScrollView>
58    </ContentPage.Content>
59 </ContentPage>
```

```
1 /*! \class The OffSocketSwitch View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the OffSocketSwitch View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OffSocketSwitch : ContentPage
20     {
21         public OffSocketSwitch()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.SocketPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Energy\OffSocketSwitch.xaml.cs
58  /** This function is called before the page is displayed and it creates an object
59  ans uses its SetLvl method to set the players level in the app
60  * and display it in the navigation bar.
61  */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The OrganicFood View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the OrganicFood View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OrganicFood : ContentPage
20     {
21         public OrganicFood()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.OrganicPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The OutsideOnce View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the OutsideOnce View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OutsideOnce : ContentPage
20     {
21         public OutsideOnce()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTwoPoints();
55                 HomePointsUpdate helper2 = new HomePointsUpdate();
56                 helper2.OutsidePoints();
57                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The OwnCoffee View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the OwnCoffee View Class. This class is the eco action that the user
8  * can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class OwnCoffee : ContentPage
20     {
21         public OwnCoffee()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTwoPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.OwnCoffeePoints();
57                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\OwnCoffee.xaml.cs
58  /* This function is called before the page is displayed and it creates an object
59  ans uses its SetLvl method to set the players level in the app
60  * and display it in the navigation bar.
61  */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The Picnic View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the Picnic View Class. This class is the eco action that the user
8  * can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class Picnic : ContentPage
20     {
21         public Picnic()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.PicnicPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The PlantABush View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the PlantABush View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PlantABush : ContentPage
20     {
21         public PlantABush()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.PlantBushPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The PlantAFlower View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the PlantAFlower View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PlantAFlower : ContentPage
20     {
21         public PlantAFlower()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.PlantFlowerPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The PlantATree View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the PlantATree View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PlantATree : ContentPage
20     {
21         public PlantATree()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.PlantTreePoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The PlantIntoHome View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the PlantIntoHome View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PlantIntoHome : ContentPage
20     {
21         public PlantIntoHome()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 HomePointsUpdate helper2 = new HomePointsUpdate();
56                 helper2.PlantsInsidePoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The PublicTransport View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the PublicTransport View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Travel
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PublicTransport : ContentPage
20     {
21         public PublicTransport()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 TravelPointsUpdate helper2 = new TravelPointsUpdate();
56                 helper2.TransportPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The PurchaseReusableWater View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the PurchaseReusableWater View Class. This class is the eco action
8  * that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class PurchaseReusableWater : ContentPage
20     {
21         public PurchaseReusableWater()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.ReWaterPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Shopping\PurchaseReusableWat...
58  /** This function creates objects and calls their methods. First the security
59  methods are called and if they return false call the points updating
60  * methods
61  */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The RainBarrel View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the RainBarrel View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class RainBarrel : ContentPage
20     {
21         public RainBarrel()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
56                 helper2.BarrelPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The ReBatteries View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ReBatteries View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Shopping
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ReBatteries : ContentPage
20     {
21         public ReBatteries()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 ShoppingPointsUpdate helper2 = new ShoppingPointsUpdate();
56                 helper2.ReBatteriesPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4         x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.ReCoffeeMug"
5         xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10            <Label Text="Reusable Cup" TextColor="White" FontSize="20"
11                FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12                HorizontalOptions="StartAndExpand"/>
13                <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14                    VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15            </StackLayout>
16
17     </NavigationPage.TitleView>
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*"/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28                     Source="{local:ImageResource
29                         Application_Green_Quake.Images.SubCategories.FD.ReCoffeeMug.jpg}"/>
30                 <StackLayout Grid.Row="1"
31                         VerticalOptions="Center"
32                         HorizontalOptions="Center"
33                         Margin="15,0,15,0"
34                         Padding="0,5,0,5"
35                         BackgroundColor="#D3D3D3">
36                     <Label Text="Use a reusable coffee mug or cup for your coffee. This
37 highly reduces waste produced from throwing away coffee cups. Waste is a large issue as most
38 of waste does not rot and therefore ends up at a landfill which keep getting bigger and
39 bigger and this cannot last forever."
40                         TextColor="Black"
41                         HorizontalTextAlignment="Center"
42                         FontSize="20"/>
43                 </StackLayout>
44                 <StackLayout Grid.Row="2"
45                         Spacing="10">
46                     <Label Text="4 POINTS!">
47                         TextColor="Black"
48                         FontAttributes="Bold"
49                         FontSize="20"
50                         HorizontalOptions="Center"/>
51                     <Button Text="Completed"
52                         BackgroundColor="#50C878"
53                         TextColor="White"
54                         Margin="60,0,60,0"
55                         Clicked="AddPointsClicked"/>
56                 </StackLayout>
57             </Grid>
58         </ScrollView>
59     </ContentPage.Content>
60 </ContentPage>
```

```
1 /*! \class The ReCoffeeMug View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ReCoffeeMug View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ReCoffeeMug : ContentPage
20     {
21         public ReCoffeeMug()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.ReCoffeeMugPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\ReCoffeeMug.xa...
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The RefrigeratorDown View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the RefrigeratorDown View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class RefrigeratorDown : ContentPage
20     {
21         public RefrigeratorDown()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.FridgePoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Energy\RefrigeratorDown.xaml.cs
58  /** This function is called before the page is displayed and it creates an object
59  ans uses its SetLvl method to set the players level in the app
60  * and display it in the navigation bar.
61  */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

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

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

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

```
1 /*! \class The ReusableWater View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ReusableWater View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ReusableWater : ContentPage
20     {
21         public ReusableWater()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
56                 helper2.ReWaterPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4         x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.SaveLeftOvers"
5         xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10            <Label Text="Save Leftovers" TextColor="White" FontSize="20"
11                FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12                HorizontalOptions="StartAndExpand"/>
13            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14                VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15        </StackLayout>
16
17    </NavigationPage.TitleView>
18    <ContentPage.Content>
19        <ScrollView>
20            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                <Grid.RowDefinitions>
22                    <RowDefinition Height="2*"/>
23                    <RowDefinition Height="2*"/>
24                    <RowDefinition Height="*"/>
25                </Grid.RowDefinitions>
26
27                <Image Aspect="AspectFill"
28                     Source="{local:ImageResource
29                         Application_Green_Quake.Images.SubCategories.FD.SaveLeftovers.jpg}" />
30
31                <StackLayout Grid.Row="1"
32                            VerticalOptions="Center"
33                            HorizontalOptions="Center"
34                            Margin="15,0,15,0"
35                            Padding="0,5,0,5"
36                            BackgroundColor="#D3D3D3">
37                    <Label Text="Instead of throwing the leftovers. Save them for later.
38 Food waste is a major concern in today world as it is at large. When food rots it emits
39 methane, a powerful greenhouse gas which is 25 times worse for the ozone layer than CO2. 60%
40 of methane is produced by humans."
41                        TextColor="Black"
42                        HorizontalTextAlignment="Center"
43                        FontSize="20" />
44                </StackLayout>
45                <StackLayout Grid.Row="2"
46                            Spacing="10">
47                    <Label Text="6 POINTS!">
48                        TextColor="Black"
49                        FontAttributes="Bold"
50                        FontSize="20"
51                        HorizontalOptions="Center" />
52                    <Button Text="Completed"
53                           BackgroundColor="#50C878"
54                           TextColor="White"
55                           Margin="60,0,60,0"
56                           Clicked="AddPointsClicked" />
57                </StackLayout>
58            </Grid>
59        </ScrollView>
60    </ContentPage.Content>
61 </ContentPage>
```

```
1 /*! \class The SaveLeftOvers View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SaveLeftOvers View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SaveLeftOvers : ContentPage
20     {
21         public SaveLeftOvers()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateBySixPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.SaveLeftOversPoints();
57                 await DisplayAlert("Points Added", AppConstants.sixPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\SaveLeftOvers.xa...
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

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

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The SealDrafts View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SealDrafts View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SealDrafts : ContentPage
20     {
21         public SealDrafts()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.SealDraftsPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The SealDucts View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SealDucts View Class. This class is the eco action that the user
8  * can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SealDucts : ContentPage
20     {
21         public SealDucts()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.SealDuctsPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The SetUpFruitGarden View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SetUpFruitGarden View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SetUpFruitGarden : ContentPage
20     {
21         public SetUpFruitGarden()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.FruitGardenPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Outdoors\SetUpFruitGarden.xaml...
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The SetUpHerbGarden View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SetUpHerbGarden View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.OutOfors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SetUpHerbGarden : ContentPage
20     {
21         public SetUpHerbGarden()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.HerbGardenPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Outdoors\SetUpHerbGarden.xaml.cs
58  /// ** This function creates objects and calls their methods. First the security
59  //methods are called and if they return false call the points updating
60  //    * methods
61  //    */
62  protected override void OnAppearing()
63  {
64      GetData data = new GetData();
65      data.SetLvl();
66
67      theLevel.Text = "LVL: " + GetData.lvl.ToString();
68  }
69 }
```

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

```
1 /*! \class The SetUpRecyclingBin View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SetUpRecyclingBin View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SetUpRecyclingBin : ContentPage
20     {
21         public SetUpRecyclingBin()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 WastePointsUpdate helper2 = new WastePointsUpdate();
56                 helper2.SetUpRecyclingBinPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Waste\SetUpRecyclingBin.xaml.cs
58  /**
59   * This function creates objects and calls their methods. First the security
60   * methods are called and if they return false call the points updating
61   */
62   protected override void OnAppearing()
63   {
64     GetData data = new GetData();
65     data.SetLvl();
66
67     theLevel.Text = "LVL: " + GetData.lvl.ToString();
68   }
69 }
```

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

```
1 /*! \class The SetUpVegetableGarden View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SetUpVegetableGarden View Class. This class is the eco action
8  * that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Outdoors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SetUpVegetableGarden : ContentPage
20     {
21         public SetUpVegetableGarden()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.VegetableGardenPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Outdoors\SetUpVegetableGarde...
58  /** This function creates objects and calls their methods. First the security
methods are called and if they return false call the points updating
   * methods
59   */
60   */
61   protected override void OnAppearing()
62   {
63     GetData data = new GetData();
64     data.SetLvl();
65
66     theLevel.Text = "LVL: " + GetData.lvl.ToString();
67   }
68 }
69 }
```

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

```
1 /*! \class The ShareThisApp View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ShareThisApp View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ShareThisApp : ContentPage
20     {
21         public ShareThisApp()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
56                 helper2.SharePoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Community\ShareThisApp.xaml.cs
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The ShowerBucket View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ShowerBucket View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ShowerBucket : ContentPage
20     {
21         public ShowerBucket()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
56                 helper2.ShowerBucketPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

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

```
58     }
59     /** This function is called before the page is displayed and it creates an object
60     ans uses its SetLvl method to set the players level in the app
61     * and display it in the navigation bar.
62     */
63     protected override void OnAppearing()
64     {
65         GetData data = new GetData();
66         data.SetLvl();
67
68         theLevel.Text = "LVL: " + GetData.lvl.ToString();
69     }
70 }
```

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

```
1 /*! \class The SolarPanel View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SolarPanel View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Energy
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SolarPanel : ContentPage
20     {
21         public SolarPanel()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 EnergyPointsUpdate helper2 = new EnergyPointsUpdate();
56                 helper2.SolarPanelPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The SpreadAwareness View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SpreadAwareness View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Community
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SpreadAwareness : ContentPage
20     {
21         public SpreadAwareness()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 CommunityPointsUpdate helper2 = new CommunityPointsUpdate();
56                 helper2.awarenessPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Community\SpreadAwareness.xa...
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The SteelStraw View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the SteelStraw View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class SteelStraw : ContentPage
20     {
21         public SteelStraw()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.SteelStrawPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\SteelStraw.xaml.cs
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4         x:Class="Application_Green_Quake.Views.EcoActions.Habits.TimedShower"
5         xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10            <Label Text="Timed Shower" TextColor="White" FontSize="20"
11                FontAttributes="Italic" VerticalOptions="CenterAndExpand"
12                HorizontalOptions="StartAndExpand"/>
13                <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14                    VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15            </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="2*"/>
23                     <RowDefinition Height="2*"/>
24                     <RowDefinition Height="*"/>
25                 </Grid.RowDefinitions>
26
27                 <Image Aspect="AspectFill"
28                     Source="{local:ImageResource
29                         Application_Green_Quake.Images.SubCategories.Habits.TimedShower.jpg}">
30                     <StackLayout Grid.Row="1"
31                         VerticalOptions="Center"
32                         HorizontalOptions="Center"
33                         Margin="15,0,15,0"
34                         Padding="0,5,0,5"
35                         BackgroundColor="#D3D3D3">
36                         <Label Text="Time your showers. A shower uses five gallons of water per
37                         minute when it does not have a water saving shower head. Therefore try spending a max of 3
38                         minutes in there.">
39                             TextColor="Black"
40                             HorizontalTextAlignment="Center"
41                             FontSize="20"/>
42                     </StackLayout>
43                     <StackLayout Grid.Row="2"
44                         Spacing="10">
45                         <Label Text="4 POINTS!">
46                             TextColor="Black"
47                             FontAttributes="Bold"
48                             FontSize="20"
49                             HorizontalOptions="Center"/>
50                         <Button Text="Completed"
51                             BackgroundColor="#50C878"
52                             TextColor="White"
53                             Margin="60,0,60,0"
54                             Clicked="AddPointsClicked"/>
55                     </StackLayout>
56                 </Grid>
57             </ScrollView>
58         </ContentPage.Content>
59     </ContentPage>
```

```
1 /*! \class The TimedShower View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the TimedShower View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Habits
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class TimedShower : ContentPage
20     {
21         public TimedShower()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
56                 helper2.TimedShowerInsteadPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The ToiletFlushes View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the ToiletFlushes View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Home
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class ToiletFlushes : ContentPage
20     {
21         public ToiletFlushes()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 HomePointsUpdate helper2 = new HomePointsUpdate();
56                 helper2.ToiletPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The TurnOffLights View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the TurnOffLights View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16
17 namespace Application_Green_Quake.Views.EcoActions.Habits
18 {
19     [XamlCompilation(XamlCompilationOptions.Compile)]
20     public partial class TurnOffLights : ContentPage
21     {
22         public TurnOffLights()
23         {
24             InitializeComponent();
25             OnAppearing();
26         }
27         /** This function creates objects and calls their methods. First the security
28         methods are called and if they return false call the points updating
29         * methods
30         */
31         private async void AddPointsClicked(object sender, EventArgs e)
32         {
33             SecurityMethods checks = new SecurityMethods();
34             Task<bool> myTask = checks.DayLimitLock();
35             await myTask;
36
37             Task<bool> myTaskTwo = checks.TimeLimitLock();
38             await myTaskTwo;
39
40             if (myTask.Result)
41             {
42                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
43 day.", "OK");
44                 await Navigation.PushAsync(new MainMenu());
45             }
46             else if (myTaskTwo.Result)
47             {
48                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
49 next Action.", "OK");
50                 await Navigation.PushAsync(new MainMenu());
51             }
52             else
53             {
54                 PointsUpdate helper = new PointsUpdate();
55                 helper.UpdateByTwoPoints();
56                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
57                 helper2.OffLightsPoints();
58                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
59                 await Navigation.PushAsync(new MainMenu());
60             }
61         }
62     }
63 }
```

```
58     }
59     /** This function is called before the page is displayed and it creates an object
60     ans uses its SetLvl method to set the players level in the app
61     * and display it in the navigation bar.
62     */
63     protected override void OnAppearing()
64     {
65         GetData data = new GetData();
66         data.SetLvl();
67
68         theLevel.Text = "LVL: " + GetData.lvl.ToString();
69     }
70 }
```

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

```
1 /*! \class The UpBirdfeeder View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the UpBirdfeeder View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.OutOfors
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class UpBirdfeeder : ContentPage
20     {
21         public UpBirdfeeder()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 OutdoorsPointsUpdate helper2 = new OutdoorsPointsUpdate();
56                 helper2.BirdFeederPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

```
1 /*! \class The UseBiogradableBinBags View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the UseBiogradableBinBags View Class. This class is the eco action
8  * that the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class UseBiogradableBinBags : ContentPage
20     {
21         public UseBiogradableBinBags()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByFourPoints();
55                 WastePointsUpdate helper2 = new WastePointsUpdate();
56                 helper2.BioBinBagPoints();
57                 await DisplayAlert("Points Added", AppConstants.fourPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Waste\UseBiogradableBinBags.x...  
58  /** This function creates objects and calls their methods. First the security  
methods are called and if they return false call the points updating  
59   * methods  
60   */  
61  protected override void OnAppearing()  
62  {  
63      GetData data = new GetData();  
64      data.SetLvl();  
65  
66      theLevel.Text = "LVL: " + GetData.lvl.ToString();  
67  }  
68 }  
69 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4         x:Class="Application_Green_Quake.Views.EcoActions.Habits.UseMatches"
5         xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10            <Label Text="Matches" TextColor="White" FontSize="20" FontAttributes="Italic"
11            VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13            VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14        </StackLayout>
15    </NavigationPage.TitleView>
16
17    <ContentPage.Content>
18        <ScrollView>
19            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                <Grid.RowDefinitions>
21                    <RowDefinition Height="2*"/>
22                    <RowDefinition Height="2*"/>
23                    <RowDefinition Height="*"/>
24                </Grid.RowDefinitions>
25
26                <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28 Application_Green_Quake.Images.SubCategories.Habits.MatchOverLighter.jpg}"/>
29
30                <StackLayout Grid.Row="1"
31                            VerticalOptions="Center"
32                            HorizontalOptions="Center"
33                            Margin="15,0,15,0"
34                            Padding="0,5,0,5"
35                            BackgroundColor="#D3D3D3">
36                    <Label Text="Use matches over lighters. Matches are biodegradable ,
37 whereas lighters are plastic and aren't recyclable in most places. The creation of lighters
38 also involves more processing and more transportation of the various components, which come
39 with their own waste and pollution."
40                    TextColor="Black"
41                    HorizontalTextAlignment="Center"
42                    FontSize="20"/>
43                </StackLayout>
44                <StackLayout Grid.Row="2"
45                            Spacing="10">
46                    <Label Text="2 POINTS!">
47                        TextColor="Black"
48                        FontAttributes="Bold"
49                        FontSize="20"
50                        HorizontalOptions="Center"/>
51                    <Button Text="Completed"
52                            BackgroundColor="#50C878"
53                            TextColor="White"
54                            Margin="60,0,60,0"
55                            Clicked="AddPointsClicked"/>
56                </StackLayout>
57            </Grid>
58        </ScrollView>
59    </ContentPage.Content>
60
61 </ContentPage>
```

```
1 /*! \class The UseMatches View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the UseMatches View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Habits
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class UseMatches : ContentPage
20     {
21         public UseMatches()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTwoPoints();
55                 HabitsPointsUpdate helper2 = new HabitsPointsUpdate();
56                 helper2.MatchesPoints();
57                 await DisplayAlert("Points Added", AppConstants.twoPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
58     /** This function is called before the page is displayed and it creates an object
59     that uses its SetLvl method to set the player's level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

```
1 /*! \class The UseRecyclingBin View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the UseRecyclingBin View Class. This class is the eco action that
8  * the user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Waste
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class UseRecyclingBin : ContentPage
20     {
21         public UseRecyclingBin()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 WastePointsUpdate helper2 = new WastePointsUpdate();
56                 helper2.RecyclingBinPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

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

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

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

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4         x:Class="Application_Green_Quake.Views.EcoActions.FoodAndDrink.WaterOverFizzy"
5         xmlns:local="clr-namespace:Application_Green_Quake.Models">
6
7     <NavigationPage.TitleView>
8         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
9             VerticalOptions="EndAndExpand" Spacing="0">
10            <Label Text="H2O" TextColor="White" FontSize="20" FontAttributes="Italic"
11                VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
12            <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
13                VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
14        </StackLayout>
15
16    </NavigationPage.TitleView>
17    <ContentPage.Content>
18        <ScrollView>
19            <Grid VerticalOptions="FillAndExpand" RowSpacing="0">
20                <Grid.RowDefinitions>
21                    <RowDefinition Height="2*"/>
22                    <RowDefinition Height="2*"/>
23                    <RowDefinition Height="*"/>
24                </Grid.RowDefinitions>
25
26                <Image Aspect="AspectFill"
27                     Source="{local:ImageResource
28                         Application_Green_Quake.Images.SubCategories.FD.WaterOverFiz.jpg}"/>
29
30                <StackLayout Grid.Row="1"
31                            VerticalOptions="Center"
32                            HorizontalOptions="Center"
33                            Margin="15,0,15,0"
34                            Padding="0,5,0,5"
35                            BackgroundColor="#D3D3D3">
36                    <Label Text="Drink water instead of other drinks. This is much healthier
37 and better for the environment. Water is the healthiest drink for the human body and is
38 better for the environment as it takes a lot more to produce fizzy drinks which emit CO2."
39                    TextColor="Black"
40                    HorizontalTextAlignment="Center"
41                    FontSize="20"/>
42                </StackLayout>
43                <StackLayout Grid.Row="2"
44                            Spacing="10">
45                    <Label Text="8 POINTS!">
46                        TextColor="Black"
47                        FontAttributes="Bold"
48                        FontSize="20"
49                        HorizontalOptions="Center"/>
50                    <Button Text="Completed"
51                        BackgroundColor="#50C878"
52                        TextColor="White"
53                        Margin="60,0,60,0"
54                        Clicked="AddPointsClicked"/>
55                </StackLayout>
56            </Grid>
57        </ScrollView>
58    </ContentPage.Content>
59 </ContentPage>
```

```
1 /*! \class The WaterOverFizzy View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the WaterOverFizzy View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.FoodAndDrink
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class WaterOverFizzy : ContentPage
20     {
21         public WaterOverFizzy()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByEightPoints();
55                 FoodAndDrinkPointsUpdate helper2 = new FoodAndDrinkPointsUpdate();
56                 helper2.WaterOverFizzyPoints();
57                 await DisplayAlert("Points Added", AppConstants.eightPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

```
4/29/2021      c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\FoodAndDrink\WaterOverFizzy.x...
58     /** This function is called before the page is displayed and it creates an object
59     ans uses its SetLvl method to set the players level in the app
60     * and display it in the navigation bar.
61     */
62     protected override void OnAppearing()
63     {
64         GetData data = new GetData();
65         data.SetLvl();
66
67         theLevel.Text = "LVL: " + GetData.lvl.ToString();
68     }
69 }
```

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

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

4/29/2021

c:\Application Green Quake\Application Green Quake\Application Green Quake\Views\EcoActions\Work\WorkingRemotely.xaml.cs

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

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

```
1 /*! \class The WSShowerHead View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the WSShowerHead View Class. This class is the eco action that the
8  * user can log.
9  */
10
11
12
13
14
15
16 namespace Application_Green_Quake.Views.EcoActions.Water
17 {
18     [XamlCompilation(XamlCompilationOptions.Compile)]
19     public partial class WSShowerHead : ContentPage
20     {
21         public WSShowerHead()
22         {
23             InitializeComponent();
24             OnAppearing();
25         }
26         /** This function creates objects and calls their methods. First the security
27         methods are called and if they return false call the points updating
28         * methods
29         */
30         private async void AddPointsClicked(object sender, EventArgs e)
31         {
32             SecurityMethods checks = new SecurityMethods();
33             Task<bool> myTask = checks.DayLimitLock();
34             await myTask;
35
36             Task<bool> myTaskTwo = checks.TimeLimitLock();
37             await myTaskTwo;
38
39             if (myTask.Result)
40             {
41                 await DisplayAlert("Daily Limit Reached", "You can only log 15 Actions per
42 day.", "OK");
43                 await Navigation.PushAsync(new MainMenu());
44             }
45             else if (myTaskTwo.Result)
46             {
47                 await DisplayAlert("Too soon", "You must wait 1 minute before logging the
48 next Action.", "OK");
49                 await Navigation.PushAsync(new MainMenu());
50             }
51             else
52             {
53                 PointsUpdate helper = new PointsUpdate();
54                 helper.UpdateByTenPoints();
55                 WaterPointsUpdate helper2 = new WaterPointsUpdate();
56                 helper2.WSSowerHeadPoints();
57                 await DisplayAlert("Points Added", AppConstants.tenPointsMsg, "OK");
58                 await Navigation.PushAsync(new MainMenu());
59             }
60         }
61     }
62 }
```

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

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <pages:PopupPage x:Class="Application_Green_Quake.Views.LeaderboardPage.LeaderBoardPopUp"
3             xmlns="http://xamarin.com/schemas/2014/forms"
4             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5             xmlns:animations="clr-
6 namespace:Rg.Plugins.Popup.Animations;assembly=Rg.Plugins.Popup"
7             xmlns:pages="clr-
8 namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
9             BackgroundColor="#002a1e">
10
11    <StackLayout HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" >
12        <Label x:Name="theName" TextColor="White" FontSize="20"
13             HorizontalTextAlignment="Center"/>
14        <Image x:Name="profileImage" HeightRequest="300"/>
15        <Label x:Name="theRank" TextColor="White" FontSize="20"
16             HorizontalTextAlignment="Center"/>
17        <Label x:Name="theBio" TextColor="White" FontSize="20"
18             HorizontalTextAlignment="Center"/>
19        <Label x:Name="thePoints" TextColor="White" FontSize="20"
20             HorizontalTextAlignment="Center"/>
21    </StackLayout>
22
23 </pages:PopupPage>
```

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

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             xmlns:models="clr-namespace:Application_Green_Quake.Models;assembly=Application
5             Green Quake"
6             xmlns:xForms="clr-
7             namespace:Syncfusion.SfPicker.XForms;assembly=Syncfusion.SfPicker.XForms"
8             x:Class="Application_Green_Quake.Views.LeaderboardPage.TopTableLeaderBoard">
9     <ContentPage.Content>
10    <StackLayout BackgroundColor="#002a1e">
11        <xForms:SfPicker x:Name="picker"
12                         ItemHeight="45"
13                         HeaderText="Filter by Country"
14                         PickerMode="Dialog"
15                         PickerHeight="350"
16                         PickerWidth="350"
17                         ShowFooter="True"
18                         OkButtonClicked="PickerOnOkButtonClicked"
19                         HeaderBackgroundColor="#50C878"
20                         SelectedItemTextColor="#50C878"
21                         OKButtonTextColor="#50C878"
22                         CancelButtonTextColor="#50C878"/>
23        <StackLayout Orientation="Horizontal" Margin="15,5,15,5">
24            <Label Text="Leader Board" FontSize="22" HorizontalTextAlignment="Center"
25             FontFamily="Proxima Nova" TextColor="White"/>
26            <Image Source="{models:ImageResource
27             Application_Green_Quake.Images.filter.png}" HorizontalOptions="EndAndExpand">
28                <Image.GestureRecognizers>
29                    <TapGestureRecognizer Tapped="ImageClicked"/>
30                </Image.GestureRecognizers>
31            </Image>
32        </StackLayout>
33        <ListView x:Name="LeaderBoard"
34                  ItemTapped="OnItemTapped"
35                  RowHeight="70">
36            <ListView.ItemTemplate>
37                <DataTemplate>
38                    <ViewCell>
39                        <StackLayout BackgroundColor="White">
40                            <Grid Margin="20,0,20,0">
41                                <Grid.ColumnDefinitions>
42                                    <ColumnDefinition Width="*"/>
43                                    <ColumnDefinition Width="2*"/>
44                                    <ColumnDefinition Width="2*"/>
45                                </Grid.ColumnDefinitions>
46                                <Grid.RowDefinitions>
47                                    <RowDefinition Height="*"/>
48                                </Grid.RowDefinitions>
49
50                                <Frame Padding="0" Margin="0,5,0,5" CornerRadius="80">
51                                    <Image Source="{Binding image}"
52                                     Aspect="AspectFill"/>
53                                </Frame>
54
55                            <StackLayout Grid.Column="1" Margin="10,5,0,0"
56                           Spacing="0">
57                                <Label Text="{Binding username}" TextColor="Black"
58                               FontAttributes="Bold" FontSize="15" FontFamily="Roboto"/>
59                                <Label Text="{Binding points, StringFormat='Score:
60 {0}'}" FontSize="12" TextColor="Silver" FontFamily="Roboto"/>
61                            </StackLayout>
62                            <StackLayout Grid.Column="2">
```

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

```
1 /*! \class The TopTabLeaderBoard View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the TopTabLeaderBoard View Class. This is the class for the
8  * leaderboard screen.
9  */
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
```

```
/*!\class The TopTabLeaderBoard View Class
* \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
* c00228956@itcarlow.ie
* \date 28/04/2021
* \section desc_sec Description
*
* Description: This is the TopTabLeaderBoard View Class. This is the class for the
leaderboard screen.
*/
using Application_Green_Quake.Models;
using Firebase.Database;
using Firebase.Database.Query;
using Firebase.Storage;
using Rg.Plugins.Popup.Services;
using System;
using System.Collections.ObjectModel;
using System.Linq;
using Acr.UserDialogs;
using Application_Green_Quake.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using SelectionChangedEventArgs = Syncfusion.SfPicker.XForms.SelectionChangedEventArgs;
namespace Application_Green_Quake.Views.LeaderboardPage
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TopTabLeaderBoard : ContentPage
    {
        private string selectedNation = "All";
        private int min = 0;
        private int count = 10;
        IAuth auth;
        public TopTabLeaderBoard()
        {
            InitializeComponent();
            ObservableCollection<object> countryNameCollection = new
ObservableCollection<object>();

// Add all the options into the filter
countryNameCollection.Add("All");
countryNameCollection.Add("Me");
countryNameCollection.Add("Albania");
countryNameCollection.Add("Andorra");
countryNameCollection.Add("Armenia");
countryNameCollection.Add("Austria");
countryNameCollection.Add("Azerbaijan");
countryNameCollection.Add("Belarus");
countryNameCollection.Add("Belgium");
countryNameCollection.Add("Bosnia and Herzegovina");
countryNameCollection.Add("Bulgaria");
countryNameCollection.Add("Croatia");
countryNameCollection.Add("Cyprus");
countryNameCollection.Add("Czech Republic");
countryNameCollection.Add("Denmark");
countryNameCollection.Add("Estonia");
countryNameCollection.Add("Finland");
countryNameCollection.Add("France");
countryNameCollection.Add("Georgia");
countryNameCollection.Add("Germany");
countryNameCollection.Add("Greece");
```

```

59         countryNameCollection.Add("Hungary");
60         countryNameCollection.Add("Iceland");
61         countryNameCollection.Add("Ireland");
62         countryNameCollection.Add("Italy");
63         countryNameCollection.Add("Kazakhstan");
64         countryNameCollection.Add("Kosovo");
65         countryNameCollection.Add("Latvia");
66         countryNameCollection.Add("Liechtenstein");
67         countryNameCollection.Add("Lithuania");
68         countryNameCollection.Add("Luxembourg");
69         countryNameCollection.Add("Malta");
70         countryNameCollection.Add("Moldova");
71         countryNameCollection.Add("Monaco");
72         countryNameCollection.Add("Montenegro");
73         countryNameCollection.Add("Netherlands");
74         countryNameCollection.Add("North Macedonia");
75         countryNameCollection.Add("Norway");
76         countryNameCollection.Add("Poland");
77         countryNameCollection.Add("Portugal");
78         countryNameCollection.Add("Romania");
79         countryNameCollection.Add("Russia");
80         countryNameCollection.Add("San Marino");
81         countryNameCollection.Add("Serbia");
82         countryNameCollection.Add("Slovakia");
83         countryNameCollection.Add("Slovenia");
84         countryNameCollection.Add("Spain");
85         countryNameCollection.Add("Sweden");
86         countryNameCollection.Add("Switzerland");
87         countryNameCollection.Add("Turkey");
88         countryNameCollection.Add("Ukraine");
89         countryNameCollection.Add("United Kingdom");
90         countryNameCollection.Add("Vatican City");

91
92         picker.ItemsSource = countryNameCollection;
93         auth = DependencyService.Get<IAuth>();
94         OnAppearing();
95     }
96     /** This function is called before the page is displayed.
97     */
98     protected override async void OnAppearing()
99     {
100         //If All gets selected from the filter then display all the profiles
101         if (selectedNation == "All")
102         {
103             UserDialogs.Instance.ShowLoading();
104             FirebaseClient firebaseClient = new FirebaseClient("https://application-
green-quake-default-rtdb.firebaseio.com/");
105
106             //Save the data into a list and order it by points
107             var list = (await firebaseClient
108                 .Child("Points")
109                 .OnceAsync<Points>()).Select(item => new Points
110                 {
111                     username = item.Object.username,
112                     points = item.Object.points,
113                 }).ToList().OrderByDescending(s => s.points);
114
115             //Save that data to a second list
116             var list2 = list.Select(item => new LeaderBoard
117             {
118                 username = item.username,

```

```

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

```

```

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

```

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

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

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

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

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

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

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

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4
5     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.AdvancedPage"
6         xmlns:local="clr-namespace:Application_Green_Quake.Models">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10            VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Advanced" TextColor="White" FontSize="20" FontAttributes="Italic"
12                VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20" FontAttributes="Italic"
14                VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand" Margin="0,0,30,0"/>
15         </StackLayout>
16     </NavigationPage.TitleView>
17
18     <ContentPage.Content>
19         <ScrollView>
20             <Grid Margin="5,5,5,5">
21                 <Grid.RowDefinitions>
22                     <RowDefinition Height="200" />
23                 </Grid.RowDefinitions>
24                 <Grid.ColumnDefinitions>
25                     <ColumnDefinition />
26                 </Grid.ColumnDefinitions>
27
28                 <Image Grid.Column="0"
29                     Grid.Row="0"
30                     Aspect="AspectFill"
31                     Source="{local:ImageResource
32 Application_Green_Quake.Images.SubCategories.Advanced.Fix.jpg}"/>
33                 <StackLayout Grid.Column="0"
34                     Grid.Row="0"
35                     BackgroundColor="Black"
36                     Opacity=".5">
37                     <StackLayout.GestureRecognizers>
38                         <TapGestureRecognizer Tapped="NavigateToFix"/>
39                     </StackLayout.GestureRecognizers>
40                 </StackLayout>
41                 <StackLayout Grid.Column="0"
42                     Grid.Row="0"
43                     VerticalOptions="End"
44                     Spacing="5"
45                     Margin="40,0,40,15">
46                     <StackLayout.GestureRecognizers>
47                         <TapGestureRecognizer Tapped="NavigateToFix"/>
48                     </StackLayout.GestureRecognizers>
49                     <Label Text="Fix It"
50                         TextColor="White"
51                         HorizontalOptions="Center"
52                         FontSize="24"
53                         FontAttributes="Bold"
54                         FontFamily="Proxima Nova"/>
55                     <Label Text="Instead of throwing it away try fix it."
56                         TextColor="White"
57                         HorizontalOptions="Center"
58                         FontSize="15"
59                         FontFamily="Proxima Nova Thin"
60                         HorizontalTextAlignment="Center"/>
61                 </StackLayout>
62             </Grid>
63         </ContentPage.Content>
64     </ContentPage>
65 
```

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

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

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

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

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

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

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

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

```
60         await Navigation.PushAsync(new ShareThisApp());
61     }
62     /** This function is called before the page is displayed and it creates an object
63     and uses its SetLvl method to set the player's level in the app
64     * and display it in the navigation bar.
65     */
66     protected override void OnAppearing()
67     {
68         GetData data = new GetData();
69         data.SetLvl();
70
71         theLevel.Text = "LVL: " + GetData.lvl.ToString();
72     }
73 }
```

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

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

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

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

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

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

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

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

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

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

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

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

```
121     {
122         await Navigation.PushAsync(new SealDucts());
123     }
124     /** This function is called before the page is displayed and it created an object
125     ans uses it's SetLvl method to set the players level in the app
126     * and display it in the navigation bar.
127     */
128     protected override void OnAppearing()
129     {
130         GetData data = new GetData();
131         data.SetLvl();
132         theLevel.Text = "LVL: " + GetData.lvl.ToString();
133     }
134 }
135 }
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
60     {
61         await Navigation.PushAsync(new UseMatches());
62     }
63     /** This function is called before the page is displayed and it created an object
64     ans uses it's SetLvl method to set the players level in the app
65     * and display it in the navigation bar.
66     */
67     protected override void OnAppearing()
68     {
69         GetData data = new GetData();
70         data.SetLvl();
71         theLevel.Text = "LVL: " + GetData.lvl.ToString();
72     }
73 }
74 }
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

414 |</ContentPage>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
57                         HorizontalOptions="Center"
58                         FontSize="15"
59                         fontFamily="Proxima Nova Thin"
60                         HorizontalTextAlignment="Center"/>
61             
```

```
62
63             <Image Grid.Column="0"
64                 Grid.Row="1"
65                 Aspect="AspectFill"
66                 Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.PublicTransport.jpg}"/>
67             
```

```
68             <StackLayout Grid.Column="0"
69                 Grid.Row="1"
70                 BackgroundColor="Black"
71                 Opacity=".5">
72                 <StackLayout.GestureRecognizers>
73                     <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
74                 </StackLayout.GestureRecognizers>
75             
```

```
76             <StackLayout Grid.Column="0"
77                 Grid.Row="1"
78                 VerticalOptions="End"
79                 Spacing="5"
80                 Margin="40,0,40,15">
81                 <StackLayout.GestureRecognizers>
82                     <TapGestureRecognizer Tapped="NavigateToPublicTransport"/>
83                 </StackLayout.GestureRecognizers>
84                 <Label Text="Public Transport"
85                     TextColor="White"
86                     HorizontalOptions="Center"
87                     FontSize="24"
88                     FontAttributes="Bold"
89                     fontFamily="Proxima Nova"/>
90                 <Label Text="Use Public Transport over a car."
91                     TextColor="White"
92                     HorizontalOptions="Center"
93                     FontSize="15"
94                     fontFamily="Proxima Nova Thin"
95                     HorizontalTextAlignment="Center"/>
96             
```

```
97             <Image Grid.Column="0"
98                 Grid.Row="2"
99                 Aspect="AspectFill"
100                Source="{local:ImageResource
Application_Green_Quake.Images.SubCategories.Travel.Walk.jpg}"/>
101            
```

```
102            <StackLayout Grid.Column="0"
103                 Grid.Row="2"
104                 BackgroundColor="Black"
105                 Opacity=".5">
106                 <StackLayout.GestureRecognizers>
107                     <TapGestureRecognizer Tapped="NavigateToWalk"/>
108                 </StackLayout.GestureRecognizers>
109             
```

```
110             <StackLayout Grid.Column="0"
111                 Grid.Row="2"
112                 VerticalOptions="End"
113                 Spacing="5"
114                 Margin="40,0,40,15">
115                 <StackLayout.GestureRecognizers>
116                     <TapGestureRecognizer Tapped="NavigateToWalk"/>
117                 </StackLayout.GestureRecognizers>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Application_Green_Quake.Views.EcoActions.EcoActionsSubMenu.WorkPage"
5     xmlns:local="clr-namespace:Application_Green_Quake.Models"
6     Title="Work Subcategories">
7
8     <NavigationPage.TitleView>
9         <StackLayout Orientation="Horizontal" HorizontalOptions="Fill"
10            VerticalOptions="EndAndExpand" Spacing="0">
11             <Label Text="Work" TextColor="White" FontSize="20" FontAttributes="Italic"
12             VerticalOptions="CenterAndExpand" HorizontalOptions="StartAndExpand"/>
13             <Label x:Name="theLevel" TextColor="White" FontSize="20"
14             FontAttributes="Italic" VerticalOptions="CenterAndExpand" HorizontalOptions="EndAndExpand"
15             Margin="0,0,30,0"/>
16         </StackLayout>
17     </NavigationPage.TitleView>
18
19     <ContentPage.Content>
20         <ScrollView>
21             <Grid Margin="5,5,5,5">
22                 <Grid.RowDefinitions>
23                     <RowDefinition Height="200" />
24                     <RowDefinition Height="200" />
25                     <RowDefinition Height="200" />
26                     <RowDefinition Height="200" />
27                     <RowDefinition Height="200" />
28                     <RowDefinition Height="200" />
29                     <RowDefinition Height="200" />
30                     <RowDefinition Height="200" />
31                 </Grid.RowDefinitions>
32                 <Grid.ColumnDefinitions>
33                     <ColumnDefinition />
34                 </Grid.ColumnDefinitions>
35
36                 <Image Grid.Column="0"
37                     Grid.Row="0"
38                     Aspect="AspectFill"
39                     Source="{local:ImageResource
40 Application_Green_Quake.Images.SubCategories.Travel.Carpool.jpg}"/>
41                 <StackLayout Grid.Column="0"
42                     Grid.Row="0"
43                     BackgroundColor="Black"
44                     Opacity=".5">
45                     <StackLayout.GestureRecognizers>
46                         <TapGestureRecognizer Tapped="NavigateToCarpool"/>
47                     </StackLayout.GestureRecognizers>
48                 </StackLayout>
49                 <StackLayout Grid.Column="0"
50                     Grid.Row="0"
51                     VerticalOptions="End"
52                     Spacing="5"
53                     Margin="40,0,40,15">
54                     <StackLayout.GestureRecognizers>
55                         <TapGestureRecognizer Tapped="NavigateToCarpool"/>
56                     </StackLayout.GestureRecognizers>
57                     <Label Text="Carpool"
58                         TextColor="White"
59                         HorizontalOptions="Center"
60                         FontSize="24"
61                         FontAttributes="Bold" />
62                 </StackLayout>
63             </Grid>
64         </ScrollView>
65     </ContentPage.Content>
66 
```

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

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

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

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

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

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

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

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

```
48 </Frame>
49
50     <Label x:Name="a21Txt" Text="Locked " TextColor="White"
51     HorizontalTextAlignment="Center" HeightRequest="60"/>
52     <Frame CornerRadius="60" HorizontalOptions="Start"
53     VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
54         <Image x:Name="a25" Source="{local:ImageResource
55         Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
56     </Frame>
57
58     <Label x:Name="a25Txt" Text="Locked " TextColor="White"
59     HorizontalTextAlignment="Center" HeightRequest="60"/>
60
61     <Frame CornerRadius="60" HorizontalOptions="Start"
62     VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
63         <Image x:Name="a29" Source="{local:ImageResource
64         Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
65     </Frame>
66
67     <Label x:Name="a29Txt" Text="Locked " TextColor="White"
68     HorizontalTextAlignment="Center" HeightRequest="60"/>
69     <Frame CornerRadius="60" HorizontalOptions="Start"
70     VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
71         <Image x:Name="a33" Source="{local:ImageResource
72         Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
73     </Frame>
74
75     <Label x:Name="a33Txt" Text="Locked " TextColor="White"
76     HorizontalTextAlignment="Center" HeightRequest="60"/>
77     <Frame CornerRadius="60" HorizontalOptions="Start"
78     VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
79         <Image x:Name="a37" Source="{local:ImageResource
80         Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
81     </Frame>
82
83     <Label x:Name="a37Txt" Text="Locked " TextColor="White"
84     HorizontalTextAlignment="Center" HeightRequest="60"/>
85     <Frame CornerRadius="60" HorizontalOptions="Start"
86     VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
87         <Image x:Name="a41" Source="{local:ImageResource
88         Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
89     </Frame>
90
91     <Label x:Name="a41Txt" Text="Locked " TextColor="White"
92     HorizontalTextAlignment="Center" HeightRequest="60"/>
93     <Frame CornerRadius="60" HorizontalOptions="Start"
94     VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
95         <Image x:Name="a45" Source="{local:ImageResource
96         Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
97     </Frame>
98
99     <Label x:Name="a45Txt" Text="Locked " TextColor="White"
100    HorizontalTextAlignment="Center" HeightRequest="60"/>
101    <Frame CornerRadius="60" HorizontalOptions="Start"
102    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
103        <Image x:Name="a49" Source="{local:ImageResource
104        Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
105    </Frame>
106
107    <Label x:Name="a49Txt" Text="Locked " TextColor="White"
108    HorizontalTextAlignment="Center" HeightRequest="60"/>
109    <Frame CornerRadius="60" HorizontalOptions="Start"
110    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
111        <Image x:Name="a53" Source="{local:ImageResource
112        Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
113    </Frame>
```

```
89 </Frame>
90
91     <Label x:Name="a53Txt" Text="Locked " TextColor="White"
92     HorizontalTextAlignment="Center" HeightRequest="60"/>
93     <Frame CornerRadius="60" HorizontalOptions="Start"
94     VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
95         <Image x:Name="a57" Source="{local:ImageResource
96 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
97     </Frame>
98
99
100    <Label x:Name="a57Txt" Text="Locked " TextColor="White"
101    HorizontalTextAlignment="Center" HeightRequest="60"/>
102    <Frame CornerRadius="60" HorizontalOptions="Start"
103    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
104        <Image x:Name="a61" Source="{local:ImageResource
105 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
106    </Frame>
107
108    <Label x:Name="a61Txt" Text="Locked " TextColor="White"
109    HorizontalTextAlignment="Center" HeightRequest="60"/>
110    <Frame CornerRadius="60" HorizontalOptions="Start"
111    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
112        <Image x:Name="a65" Source="{local:ImageResource
113 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
114    </Frame>
115
116    <Label x:Name="a65Txt" Text="Locked " TextColor="White"
117    HorizontalTextAlignment="Center" HeightRequest="60"/>
118    <Frame CornerRadius="60" HorizontalOptions="Start"
119    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
120        <Image x:Name="a69" Source="{local:ImageResource
121 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
122    </Frame>
123
124    <Label x:Name="a69Txt" Text="Locked " TextColor="White"
125    HorizontalTextAlignment="Center" HeightRequest="60"/>
126    <Frame CornerRadius="60" HorizontalOptions="Start"
127    VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
128        <Image x:Name="a73" Source="{local:ImageResource
129 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
130    </Frame>
131
132    <Label x:Name="a73Txt" Text="Locked " TextColor="White"
133    HorizontalTextAlignment="Center" HeightRequest="60"/>
134    </StackLayout>
135
136    <StackLayout Grid.Column="1" Grid.Row="0" VerticalOptions="Start">
137        <Frame CornerRadius="60" HorizontalOptions="Start"
138        VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b" >
```

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

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

```
213
214             <Label x:Name="a66Txt" Text="Locked " TextColor="White"
215             HorizontalTextAlignment="Center" HeightRequest="60"/>
216             <Frame CornerRadius="60" HorizontalOptions="Start"
217             VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
218                 <Image x:Name="a70" Source="{local:ImageResource
219                 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
220             </Frame>
221
222             <Label x:Name="a70Txt" Text="Locked " TextColor="White"
223             HorizontalTextAlignment="Center" HeightRequest="60"/>
224             <Frame CornerRadius="60" HorizontalOptions="Start"
225             VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
226                 <Image x:Name="a74" Source="{local:ImageResource
227                 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
228             </Frame>
229
230             <Label x:Name="a74Txt" Text="Locked " TextColor="White"
231             HorizontalTextAlignment="Center" HeightRequest="60"/>
232             <Frame CornerRadius="60" HorizontalOptions="Start"
233             VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
234                 <Image x:Name="a78" Source="{local:ImageResource
235                 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
236             </Frame>
237
238             <Label x:Name="a78Txt" Text="Locked " TextColor="White"
239             HorizontalTextAlignment="Center" HeightRequest="60"/>
240         </StackLayout>
241
242             <StackLayout Grid.Column="2" Grid.Row="0" VerticalOptions="Start">
243                 <Frame CornerRadius="60" HorizontalOptions="Start"
244                 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
245                     <Image x:Name="a3" Source="{local:ImageResource
246                     Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
247                 </Frame>
248
249                 <Label x:Name="a3Txt" Text="Locked " TextColor="White"
250                 HorizontalTextAlignment="Center" HeightRequest="60"/>
251                 <Frame CornerRadius="60" HorizontalOptions="Start"
252                 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
253                     <Image x:Name="a7" Source="{local:ImageResource
254                     Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
255                 </Frame>
256
257                 <Label x:Name="a7Txt" Text="Locked " TextColor="White"
258                 HorizontalTextAlignment="Center" HeightRequest="60"/>
259                 <Frame CornerRadius="60" HorizontalOptions="Start"
260                 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
261                     <Image x:Name="a11" Source="{local:ImageResource
262                     Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
263                 </Frame>
264
265                 <Label x:Name="a11Txt" Text="Locked " TextColor="White"
266                 HorizontalTextAlignment="Center" HeightRequest="60"/>
267                 <Frame CornerRadius="60" HorizontalOptions="Start"
268                 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
269                     <Image x:Name="a15" Source="{local:ImageResource
270                     Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
271                 </Frame>
272
273                 <Label x:Name="a15Txt" Text="Locked " TextColor="White"
274                 HorizontalTextAlignment="Center" HeightRequest="60"/>
275                 <Frame CornerRadius="60" HorizontalOptions="Start"
276                 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
277                     <Image x:Name="a19" Source="{local:ImageResource
```

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

```
295             </Frame>
296
297             <Label x:Name="a51Txt" Text="Locked " TextColor="White"
298 HorizontalTextAlignment="Center" HeightRequest="60"/>
299             <Frame CornerRadius="60" HorizontalOptions="Start"
300 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
301                 <Image x:Name="a55" Source="{local:ImageResource
302 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
303             </Frame>
304
305             <Label x:Name="a55Txt" Text="Locked " TextColor="White"
306 HorizontalTextAlignment="Center" HeightRequest="60"/>
307             <Frame CornerRadius="60" HorizontalOptions="Start"
308 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
309                 <Image x:Name="a59" Source="{local:ImageResource
310 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
311             </Frame>
312
313             <Label x:Name="a59Txt" Text="Locked " TextColor="White"
314 HorizontalTextAlignment="Center" HeightRequest="60"/>
315             <Frame CornerRadius="60" HorizontalOptions="Start"
316 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
317                 <Image x:Name="a63" Source="{local:ImageResource
318 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
319             </Frame>
320
321             <Label x:Name="a63Txt" Text="Locked " TextColor="White"
322 HorizontalTextAlignment="Center" HeightRequest="60"/>
323             <Frame CornerRadius="60" HorizontalOptions="Start"
324 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
325                 <Image x:Name="a67" Source="{local:ImageResource
326 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
327             </Frame>
328
329             <Label x:Name="a67Txt" Text="Locked " TextColor="White"
330 HorizontalTextAlignment="Center" HeightRequest="60"/>
331             <Frame CornerRadius="60" HorizontalOptions="Start"
332 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
333                 <Image x:Name="a71" Source="{local:ImageResource
334 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
335             </Frame>
336
337             <Label x:Name="a71Txt" Text="Locked " TextColor="White"
338 HorizontalTextAlignment="Center" HeightRequest="60"/>
339             </StackLayout>
340
341             <StackLayout Grid.Column="3" Grid.Row="0" VerticalOptions="Start">
342                 <Frame CornerRadius="60" HorizontalOptions="Start"
343 VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
```

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

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

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

```
1 /*! \class The Achievements View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the Achievements View Class. This class is the class that displays a
8  * Achievements on the Achievements page.
9  */
10
11 using Application_Green_Quake.ViewModels;
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.ProfilePage
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class Achievements : ContentPage
19     {
20         IAuth auth;
21
22         public Achievements()
23         {
24             InitializeComponent();
25             auth = DependencyService.Get<IAuth>();
26             OnAppearing();
27         }
28         /** This function is called before the page is displayed. It displays the images as
29         * are met
30         */
31         protected override void OnAppearing()
32         {
33             if (GetAchievementsData.brushingCount >= 5 && GetAchievementsData.brushingCount
34             {
35                 a1.Source =
36                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingBronze.");
37                 a1Txt.Text = "Brushing Bronze";
38             }
39             else if (GetAchievementsData.brushingCount >= 15 && GetAchievementsData.brushing
40             {
41                 a1.Source =
42                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingSilver.");
43                 a1Txt.Text = "Brushing Silver";
44             }
45             else if (GetAchievementsData.brushingCount >= 25)
46             {
47                 a1.Source =
48                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.brushingGold.");
49                 a1Txt.Text = "Brushing Gold";
50             }
51             if (GetAchievementsData.fullWasherCount >= 5 && GetAchievementsData.fullWasherCo
52             {
53                 a2.Source =
54                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterBronz
55                 a2Txt.Text = "Dish Washer Bronze";
56             }
57             else if (GetAchievementsData.fullWasherCount >= 15 && GetAchievementsData.fullWa
58             {
59                 a2.Source =
60                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterSilve
61                 a2Txt.Text = "Dish Washer Silver";
62             }
63         }
64     }
65 }
```

```
55     }
56     else if (GetAchievementsData.fullWasherCount >= 25)
57     {
58         a2.Source =
59         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.dishwasterGold.");
60         a2Txt.Text = "Dish Washer Gold";
61     }
62     if (GetAchievementsData.shareCount >= 5 && GetAchievementsData.shareCount < 15)
63     {
64         a3.Source =
65         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerBronze.png");
66         a3Txt.Text = "Shower Bronze";
67     }
68     else if (GetAchievementsData.shareCount >= 15 && GetAchievementsData.shareCount < 25)
69     {
70         a3.Source =
71         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerSilver.png");
72         a3Txt.Text = "Shower Silver";
73     }
74     else if (GetAchievementsData.shareCount >= 25)
75     {
76         a3.Source =
77         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.showerGold.png");
78         a3Txt.Text = "Shower Gold";
79     }
80
81     if (GetAchievementsData.timedShowerCount >= 5 && GetAchievementsData.timedShowerCount < 15)
82     {
83         a4.Source =
84         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerBronze.png");
85         a4Txt.Text = "Timed Shower Bronze";
86     }
87     else if (GetAchievementsData.timedShowerCount >= 15 && GetAchievementsData.timedShowerCount < 25)
88     {
89         a4.Source =
90         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerSilver.png");
91         a4Txt.Text = "Timed Shower Silver";
92     }
93     else if (GetAchievementsData.timedShowerCount >= 25)
94     {
95         a4.Source =
96         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.timeShowerGold.png");
97         a4Txt.Text = "Timed Shower Gold";
98     }
99
100    if (GetAchievementsData.offLigtsCount >= 5 && GetAchievementsData.offLigtsCount < 15)
101    {
102        a5.Source =
103        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightBronze.png");
104        a5Txt.Text = "Off Lights Bronze";
105    }
106    else if (GetAchievementsData.offLigtsCount >= 15 && GetAchievementsData.offLigtsCount < 25)
107    {
108        a5.Source =
109        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightSilver.png");
110        a5Txt.Text = "Off Lights Silver";
111    }
112    else if (GetAchievementsData.offLigtsCount >= 25)
113    {
114        a5.Source =
115        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Habits.lightGold.png");
116    }
```

```

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

```

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

```
211     else if (GetAchievementsData.ownCoffeeCount >= 15 && GetAchievementsData.ownCoff
212     {
213         a12.Source =
214             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.ownCoffeeSilver.png");
215             a12Txt.Text = "Own Coffee Silver";
216     }
217     else if (GetAchievementsData.ownCoffeeCount >= 25)
218     {
219         a12.Source =
220             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.ownCoffeeGold.png");
221             a12Txt.Text = "Own Coffee Gold";
222     }
223
224     if (GetAchievementsData.reCoffeeMugCount >= 5 && GetAchievementsData.reCoffeeMug
225     {
226         a13.Source =
227             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.reCupBronze.png");
228             a13Txt.Text = "Reusable Mug Bronze";
229     }
230     else if (GetAchievementsData.reCoffeeMugCount >= 15 && GetAchievementsData.reCof
231     25)
232     {
233         a13.Source =
234             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.reCupSilver.png");
235             a13Txt.Text = "Reusable Mug Silver";
236     }
237
238     if (GetAchievementsData.steelStrawCount >= 5 && GetAchievementsData.steelStrawCo
239     {
240         a14.Source =
241             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawBronze.png");
242             a14Txt.Text = "Steel Straw Bronze";
243     }
244     else if (GetAchievementsData.steelStrawCount >= 15 && GetAchievementsData.steels
245     {
246         a14.Source =
247             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawSilver.png");
248             a14Txt.Text = "Steel Straw Silver";
249     }
250     else if (GetAchievementsData.steelStrawCount >= 25)
251     {
252         a14.Source =
253             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.strawGold.png");
254             a14Txt.Text = "Steel Straw Gold";
255     }
256
257     if (GetAchievementsData.waterOverFizzyCount >= 5 && GetAchievementsData.waterOve
15)
258     {
259         a15.Source =
260             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterBronze.png");
261             a15Txt.Text = "Water Bronze";
262     }
263     else if (GetAchievementsData.waterOverFizzyCount >= 15 &&
GetAchievementsData.waterOverFizzyCount < 25)
264     {
265         a15.Source =
```

```
262     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterSilver.png");
263         a15Txt.Text = "Water Silver";
264     }
265     else if (GetAchievementsData.waterOverFizzyCount >= 25)
266     {
267         a15.Source =
268             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.FD.waterGold.png");
269             a15Txt.Text = "Water Gold";
270     }
271
272     if (GetAchievementsData.fullDryerCount >= 5 && GetAchievementsData.fullDryerCount <
273     {
274         a16.Source =
275             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerBronze.png");
276                 a16Txt.Text = "Dryer Bronze";
277     }
278     else if (GetAchievementsData.fullDryerCount >= 15 && GetAchievementsData.fullDryerCount <
279     {
280         a16.Source =
281             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerSilver.png");
282                 a16Txt.Text = "Dryer Silver";
283     }
284     else if (GetAchievementsData.fullDryerCount >= 25)
285     {
286         a16.Source =
287             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.dryerGold.png");
288                 a16Txt.Text = "Dryer Gold";
289     }
290
291     if (GetAchievementsData.hangDryCount >= 5 && GetAchievementsData.hangDryCount <
292     {
293         a17.Source =
294             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangBronze.png");
295                 a17Txt.Text = "Hang Dry Bronze";
296     }
297     else if (GetAchievementsData.hangDryCount >= 15 && GetAchievementsData.hangDryCount <
298     {
299         a17.Source =
300             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangSilver.png");
301                 a17Txt.Text = "Hang Dry Silver";
302     }
303     else if (GetAchievementsData.hangDryCount >= 25)
304     {
305         a17.Source =
306             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.hangGold.png");
307                 a17Txt.Text = "Hang Dry Gold";
308     }
309
310     if (GetAchievementsData.microwaveCount >= 5 && GetAchievementsData.microwaveCount <
311     {
312         a18.Source =
313             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveBronze.png");
314                 a18Txt.Text = "Microwave Bronze";
315     }
316     else if (GetAchievementsData.microwaveCount >= 15 && GetAchievementsData.microwaveCount <
317     {
318         a18.Source =
319             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveSilver.png");
320                 a18Txt.Text = "Microwave Silver";
321     }
322     else if (GetAchievementsData.microwaveCount >= 25)
323     {
324         a18.Source =
```

```
315     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.microwaveGold.png");
316     a18Txt.Text = "Microwave Gold";
317 }
318 if (GetAchievementsData.reBatteriesCount >= 5 && GetAchievementsData.reBatteriesCount < 15)
319 {
320     a19.Source =
321     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesBronze.png");
322     a19Txt.Text = "Reusable Batteries Bronze";
323 }
324 else if (GetAchievementsData.reBatteriesCount >= 15 && GetAchievementsData.reBatteriesCount < 25)
325 {
326     a19.Source =
327     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesSilver.png");
328     a19Txt.Text = "Reusable Batteries Silver";
329 }
330 else if (GetAchievementsData.reBatteriesCount >= 25)
331 {
332     a19.Source =
333     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.reBatteriesGold.png");
334     a19Txt.Text = "Reusable Batteries Gold";
335 }
336
337 if (GetAchievementsData.offSocketCount >= 5 && GetAchievementsData.offSocketCount < 15)
338 {
339     a20.Source =
340     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.socketBronze.png");
341     a20Txt.Text = "Socket Off Bronze";
342 }
343 else if (GetAchievementsData.offSocketCount >= 15 && GetAchievementsData.offSocketCount < 25)
344 {
345     a20.Source =
346     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.socketSilver.png");
347     a20Txt.Text = "Socket Off Silver";
348 }
349
350 if (GetAchievementsData.fullWasherCount >= 5 && GetAchievementsData.fullWasherCount < 15)
351 {
352     a21.Source =
353     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineBronze.png");
354     a21Txt.Text = "Washing Machine Bronze";
355 }
356 else if (GetAchievementsData.fullWasherCount >= 15 && GetAchievementsData.fullWasherCount < 25)
357 {
358     a21.Source =
359     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineSilver.png");
360     a21Txt.Text = "Washing Machine Silver";
361 }
362 else if (GetAchievementsData.fullWasherCount >= 25)
363 {
364     a21.Source =
365     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.washingMachineGold.png");
366     a21Txt.Text = "Washing Machine Gold";
367 }
368
369 if (GetAchievementsData.carpoolCount >= 5 && GetAchievementsData.carpoolCount < 15)
370 {
371     a22.Source =
372     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.carpoolBronze.png");
373     a22Txt.Text = "Carpool Bronze";
374 }
375 else if (GetAchievementsData.carpoolCount >= 15 && GetAchievementsData.carpoolCount < 25)
376 {
377     a22.Source =
378     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.carpoolSilver.png");
379     a22Txt.Text = "Carpool Silver";
380 }
381 else if (GetAchievementsData.carpoolCount >= 25)
382 {
383     a22.Source =
384     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Energy.carpoolGold.png");
385     a22Txt.Text = "Carpool Gold";
386 }
```

```
367     {
368         a22.Source =
369         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolBronze.p
370             a22Txt.Text = "Carpool Bronze";
371     }
372     else if (GetAchievementsData.carpoolCount >= 15 && GetAchievementsData.carpoolCo
373     {
374         a22.Source =
375         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolSilver.p
376             a22Txt.Text = "Carpool Silver";
377     }
378     else if (GetAchievementsData.carpoolCount >= 25)
379     {
380         a22.Source =
381         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.carpoolGold.png
382             a22Txt.Text = "Carpool Gold";
383     }
384
385     if (GetAchievementsData.cycleCount >= 5 && GetAchievementsData.cycleCount < 15)
386     {
387         a23.Source =
388         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleBronze.png
389             a23Txt.Text = "Cycle Bronze";
390     }
391     else if (GetAchievementsData.cycleCount >= 15 && GetAchievementsData.cycleCount
392     {
393         a23.Source =
394         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleSilver.png
395             a23Txt.Text = "Cycle Silver";
396     }
397     else if (GetAchievementsData.cycleCount >= 25)
398     {
399         a23.Source =
400         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.cycleGold.png")
401             a23Txt.Text = "Cycle Gold";
402     }
403
404     if (GetAchievementsData.ecoCarCount >= 5 && GetAchievementsData.ecoCarCount < 15)
405     {
406         a24.Source =
407         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarBronze.png"
408             a24Txt.Text = "Eco Car Bronze";
409     }
410     else if (GetAchievementsData.ecoCarCount >= 15 && GetAchievementsData.ecoCarCoun
411     {
412         a24.Source =
413         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarSilver.png"
414             a24Txt.Text = "Eco Car Silver";
415     }
416     else if (GetAchievementsData.ecoCarCount >= 25)
417     {
418         a24.Source =
419         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.eCarGold.png");
420             a24Txt.Text = "Eco Car Gold";
421     }
422
423
424     if (GetAchievementsData.transportCount >= 5 && GetAchievementsData.transportCoun
425     {
426         a25.Source =
427         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbBronze.png");
428             a25Txt.Text = "Public Transport Bronze";
429     }
430     else if (GetAchievementsData.transportCount >= 15 && GetAchievementsData.transpo
```

```
420    {
421        a25.Source =
422        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbSilver.png");
423        a25Txt.Text = "Public Transport Silver";
424    }
425    else if (GetAchievementsData.transportCount >= 25)
426    {
427        a25.Source =
428        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.pbGold.png");
429        a25Txt.Text = "Public Transport Gold";
430    }
431
432    if (GetAchievementsData.walkCount >= 5 && GetAchievementsData.walkCount < 15)
433    {
434        a26.Source =
435        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkBronze.png");
436        a26Txt.Text = "Walk Bronze";
437    }
438    else if (GetAchievementsData.walkCount >= 15 && GetAchievementsData.walkCount <
439    {
440        a26.Source =
441        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkSilver.png");
442        a26Txt.Text = "Walk Silver";
443    }
444    else if (GetAchievementsData.walkCount >= 25)
445    {
446        a26.Source =
447        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Travel.walkGold.png");
448        a26Txt.Text = "Walk Gold";
449    }
450
451    if (GetAchievementsData.applianceCount >= 5 && GetAchievementsData.applianceCoun
452    {
453        a27.Source =
454        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceBron
455        a27Txt.Text = "Eco Appliance Bronze";
456    }
457    else if (GetAchievementsData.applianceCount >= 15 && GetAchievementsData.applian
458    {
459        a27.Source =
460        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceSilv
461        a27Txt.Text = "Eco Appliance Silver";
462    }
463    else if (GetAchievementsData.applianceCount >= 25)
464    {
465        a27.Source =
466        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.applianceGold
467        a27Txt.Text = "Eco Appliance Gold";
468    }
469
470    if (GetAchievementsData.foodCount >= 5 && GetAchievementsData.foodCount < 15)
471    {
472        a28.Source =
473        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkBronze.bn
474        a28Txt.Text = "Food In Bulk Bronze";
475    }
476    else if (GetAchievementsData.foodCount >= 15 && GetAchievementsData.foodCount <
477    {
478        a28.Source =
479        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkSilver.bn
480        a28Txt.Text = "Food In Bulk Silver";
481    }
482    else if (GetAchievementsData.foodCount >= 25)
```

```
473     {
474         a28.Source =
475             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.bulkGold.png");
476             a28Txt.Text = "Food In Bulk Gold";
477     }
478
479     if (GetAchievementsData.clothesCount >= 5 && GetAchievementsData.clothesCount <
480     {
481         a29.Source =
482             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalBronze");
483             a29Txt.Text = "Ethical Clothing Bronze";
484     }
485     else if (GetAchievementsData.clothesCount >= 15 && GetAchievementsData.clothesCo
486     {
487         a29.Source =
488             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalSilver");
489             a29Txt.Text = "Ethical Clothing Silver";
490     }
491     else if (GetAchievementsData.clothesCount >= 25)
492     {
493         a29.Source =
494             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.ethicalGold.p
495             a29Txt.Text = "Ethical Clothing Gold";
496     }
497
498     if (GetAchievementsData.localCount >= 5 && GetAchievementsData.localCount < 15)
499     {
500         a30.Source =
501             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localBronze.p
502             a30Txt.Text = "Buy Local Bronze";
503     }
504     else if (GetAchievementsData.localCount >= 15 && GetAchievementsData.localCount
505     {
506         a30.Source =
507             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localSilver.p
508             a30Txt.Text = "Buy Local Silver";
509     }
510     else if (GetAchievementsData.localCount >= 25)
511     {
512         a30.Source =
513             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.localGold.png
514             a30Txt.Text = "Buy Local Gold";
515
516         if (GetAchievementsData.clothNapkinCount >= 5 && GetAchievementsData.clothNapkin
517         {
518             a31.Source =
519                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinBronze.
520                     a31Txt.Text = "Napkin Bronze";
521         }
522         else if (GetAchievementsData.clothNapkinCount >= 15 && GetAchievementsData.cloth
523             25)
524             {
525                 a31.Source =
526                     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinSilver.
527                         a31Txt.Text = "Napkin Silver";
528             }
529             else if (GetAchievementsData.clothNapkinCount >= 25)
530             {
531                 a31.Source =
532                     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.napkinGold.bn
533                         a31Txt.Text = "Napkin Gold";
534             }
535 }
```

```
525
526     if (GetAchievementsData.productCount >= 5 && GetAchievementsData.productCount <
527     {
528         a32.Source =
529         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productBronze");
530         a32Txt.Text = "Eco Product Bronze";
531     }
532     else if (GetAchievementsData.productCount >= 15 && GetAchievementsData.productCo
533     {
534         a32.Source =
535         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productSilver");
536         a32Txt.Text = "Eco Product Silver";
537     }
538     else if (GetAchievementsData.productCount >= 25)
539     {
540         a32.Source =
541         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.productGold.p
542         a32Txt.Text = "Eco Product Gold";
543     }
544
545     if (GetAchievementsData.reBagCount >= 5 && GetAchievementsData.reBagCount < 15)
546     {
547         a33.Source =
548         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagBronze.p
549         a33Txt.Text = "Reusable Bag Bronze";
550     }
551     else if (GetAchievementsData.reBagCount >= 15 && GetAchievementsData.reBagCount
552     {
553         a33.Source =
554         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagSilver.p
555         a33Txt.Text = "Reusable Bag Silver";
556     }
557     else if (GetAchievementsData.reBagCount >= 25)
558     {
559         a33.Source =
560         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.reBagGold.png
561         a33Txt.Text = "Reusable Bag Gold";
562     }
563
564     if (GetAchievementsData.looseLeafCount >= 5 && GetAchievementsData.looseLeafCoun
565     {
566         a34.Source =
567         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaBronze.png
568         a34Txt.Text = "Loose Leaf Bronze";
569     }
570     else if (GetAchievementsData.looseLeafCount >= 15 && GetAchievementsData.looseLe
571     {
572         a34.Source =
573         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaSilver.png
574         a34Txt.Text = "Loose Leaf Silver";
575     }
576     else if (GetAchievementsData.looseLeafCount >= 25)
577     {
578         a34.Source =
579         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.teaGold.png")
580         a34Txt.Text = "Loose Leaf Gold";
581     }
582
583     if (GetAchievementsData.toothbrushCount >= 5 && GetAchievementsData.toothbrushCo
584     {
585         a35.Source =
586         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushBro
587         a35Txt.Text = "Eco Brush Bronze";
```

```
578     }
579     else if (GetAchievementsData.toothbrushCount >= 15 && GetAchievementsData.toothb
580     {
581         a35.Source =
582             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushSil
583                 a35Txt.Text = "Eco Brush Silver";
584             }
585             else if (GetAchievementsData.toothbrushCount >= 25)
586             {
587                 a35.Source =
588                     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.toothbrushGol
589                         a35Txt.Text = "Eco Brush Gold";
590                     }
591
592         if (GetAchievementsData.clothTowelCount >= 5 && GetAchievementsData.clothTowelCo
593         {
594             a36.Source =
595                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelBronze.p
596                     a36Txt.Text = "Towel Bronze";
597             }
598             else if (GetAchievementsData.clothTowelCount >= 15 && GetAchievementsData.clothT
599             {
600                 a36.Source =
601                     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelSilver.p
602                         a36Txt.Text = "Towel Silver";
603                     }
604                     else if (GetAchievementsData.clothTowelCount >= 25)
605                     {
606                         a36.Source =
607                             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Shopping.towelGold.png
608                                 a36Txt.Text = "Towel Gold";
609                             }
610
611         if (GetAchievementsData.reWaterCount >= 5 && GetAchievementsData.reWaterCount <
612         {
613             a37.Source =
614                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleBronze.png
615                     a37Txt.Text = "Reusable Bottle Bronze";
616             }
617             else if (GetAchievementsData.reWaterCount >= 15 && GetAchievementsData.reWaterCo
618             {
619                 a37.Source =
620                     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleSilver.png
621                         a37Txt.Text = "Reusable Bottle Silver";
622             }
623             else if (GetAchievementsData.reWaterCount >= 25)
624             {
625                 a37.Source =
626                     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bottleGold.png")
627                         a37Txt.Text = "Reusable Bottle Gold";
628                     }
629
630         if (GetAchievementsData.showerBucketCount >= 5 && GetAchievementsData.showerBuck
631         {
632             a38.Source =
633                 ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketBronze.png
634                     a38Txt.Text = "Shower Bucket Bronze";
635             }
636             else if (GetAchievementsData.showerBucketCount >= 15 && GetAchievementsData.show
637                 25)
638             {
639                 a38.Source =
640                     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketSilver.png
```

```
630         a38Txt.Text = "Shower Bucket Silver";
631     }
632     else if (GetAchievementsData.showerBucketCount >= 25)
633     {
634         a38.Source =
635         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Water.bucketGold.png")
636         a38Txt.Text = "Shower Bucket Gold";
637     }
638
639     if (GetAchievementsData.airOutCount >= 5 && GetAchievementsData.airOutCount < 15)
640     {
641         a39.Source =
642         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airBronze.png");
643         a39Txt.Text = "Ait Out Bronze";
644     }
645     else if (GetAchievementsData.airOutCount >= 15 && GetAchievementsData.airOutCount < 25)
646     {
647         a39.Source =
648         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airSilver.png");
649         a39Txt.Text = "Ait Out Silver";
650     }
651     else if (GetAchievementsData.airOutCount >= 25)
652     {
653         a39.Source =
654         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.airGold.png");
655         a39Txt.Text = "Ait Out Gold";
656     }
657
658     if (GetAchievementsData.toiletFlushCount >= 5 && GetAchievementsData.toiletFlushCount < 15)
659     {
660         a40.Source =
661         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushBronze.png");
662         a40Txt.Text = "Save Flush Bronze";
663     }
664     else if (GetAchievementsData.toiletFlushCount >= 15 && GetAchievementsData.toiletFlushCount < 25)
665     {
666         a40.Source =
667         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushSilver.png");
668         a40Txt.Text = "Save Flush Silver";
669     }
670     else if (GetAchievementsData.toiletFlushCount >= 25)
671     {
672         a40.Source =
673         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.flushGold.png");
674         a40Txt.Text = "Save Flush Gold";
675     }
676
677     if (GetAchievementsData.nonHarmCount >= 5 && GetAchievementsData.nonHarmCount < 15)
678     {
679         a41.Source =
680         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.nonHarmfulBronze.png");
681         a41Txt.Text = "Non Harmful Bronze";
682     }
683     else if (GetAchievementsData.nonHarmCount >= 15 && GetAchievementsData.nonHarmCount < 25)
684     {
685         a41.Source =
686         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Home.nonHarmfulSilver.png");
687         a41Txt.Text = "Non Harmful Silver";
688     }
689     else if (GetAchievementsData.nonHarmCount >= 25)
690     {
691         a41.Source =
```

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

```
735     {
736         a45.Source =
737         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingBronze");
738         a45Txt.Text = "Camping Bronze";
739     }
740     else if (GetAchievementsData.campingCount >= 15 && GetAchievementsData.campingCo
741     {
742         a45.Source =
743         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingSilver");
744         a45Txt.Text = "Camping Silver";
745     }
746     else if (GetAchievementsData.campingCount >= 25)
747     {
748         a45.Source =
749         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.campingGold.p
750         a45Txt.Text = "Camping Gold";
751     }
752
753     if (GetAchievementsData.plantFlowerCount >= 5 && GetAchievementsData.plantFlower
754     {
755         a46.Source =
756         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerBronze.");
757         a46Txt.Text = "Flower Planting Bronze";
758     }
759     else if (GetAchievementsData.plantFlowerCount >= 15 && GetAchievementsData.plant
760     25)
761     {
762         a46.Source =
763         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerSilver.");
764         a46Txt.Text = "Flower Planting Silver";
765     }
766     else if (GetAchievementsData.plantFlowerCount >= 25)
767     {
768         a46.Source =
769         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.flowerGold.pn
770         a46Txt.Text = "Flower Planting Gold";
771     }
772
773     if (GetAchievementsData.picnicCount >= 5 && GetAchievementsData.picnicCount < 15
774     {
775         a47.Source =
776         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicBronze.");
777         a47Txt.Text = "Picnic Bronze";
778     }
779     else if (GetAchievementsData.picnicCount >= 15 && GetAchievementsData.picnicCoun
780     {
781         a47.Source =
782         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicSilver.");
783         a47Txt.Text = "Picnic Silver";
784     }
785     else if (GetAchievementsData.picnicCount >= 25)
786     {
787         a47.Source =
788         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.picnicGold.pn
789         a47Txt.Text = "Picnic Gold";
790     }
791
792     if (GetAchievementsData.scoopCount >= 5 && GetAchievementsData.scoopCount < 15)
793     {
794         a48.Source =
795         ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Outdoors.scoopBronze.p
796         a48Txt.Text = "Scoop Poop Bronze";
797     }
```

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

```
840     else if (GetAchievementsData.donateCount >= 25)
841     {
842         a51.Source =
843             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Community.donateGold.png");
844             a51Txt.Text = "Donations Gold";
845     }
846
847     if (GetAchievementsData.bioBinBagsCount >= 5 && GetAchievementsData.bioBinBagsCount < 15)
848     {
849         a52.Source =
850             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsBronze.png");
851             a52Txt.Text = "Bio Bin Bag Bronze";
852     }
853     else if (GetAchievementsData.bioBinBagsCount >= 15 && GetAchievementsData.bioBinBagsCount < 25)
854     {
855         a52.Source =
856             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsSilver.png");
857             a52Txt.Text = "Bio Bin Bag Silver";
858     }
859     else if (GetAchievementsData.bioBinBagsCount >= 25)
860     {
861         a52.Source =
862             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.bagsGold.png");
863             a52Txt.Text = "Bio Bin Bag Gold";
864     }
865
866     if (GetAchievementsData.billsCount >= 5 && GetAchievementsData.billsCount < 15)
867     {
868         a53.Source =
869             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsBronze.png");
870             a53Txt.Text = "Online Bills Bronze";
871     }
872     else if (GetAchievementsData.billsCount >= 15 && GetAchievementsData.billsCount < 25)
873     {
874         a53.Source =
875             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsSilver.png");
876             a53Txt.Text = "Online Bills Silver";
877     }
878     else if (GetAchievementsData.billsCount >= 25)
879     {
880         a53.Source =
881             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.billsGold.png");
882             a53Txt.Text = "Online Bills Gold";
883     }
884
885     if (GetAchievementsData.recyclingBinCount >= 5 && GetAchievementsData.recyclingBinCount < 15)
886     {
887         a54.Source =
888             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binBronze.png");
889             a54Txt.Text = "Recycling Bronze";
890     }
891     else if (GetAchievementsData.recyclingBinCount >= 15 && GetAchievementsData.recyclingBinCount < 25)
892     {
893         a54.Source =
894             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binSilver.png");
895             a54Txt.Text = "Recycling Silver";
896     }
897     else if (GetAchievementsData.recyclingBinCount >= 25)
898     {
899         a54.Source =
900             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Waste.binGold.png");
901             a54Txt.Text = "Recycling Gold";
```

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

```
944         a58.Source =
945     ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixBronze.png"
946                             a58Txt.Text = "Fixed Bronze";
947             }
948             else if (GetAchievementsData.fixCount >= 15 && GetAchievementsData.fixCount < 25
949             {
950                 a58.Source =
951             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixSilver.png"
952                             a58Txt.Text = "Fixed Silver";
953             }
954             else if (GetAchievementsData.fixCount >= 25)
955             {
956                 a58.Source =
957             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.Advanced.fixGold.png")
958                             a58Txt.Text = "Fixed Gold";
959             }
960
961             if (GetAchievementsData.efficientThermostatCount >= 1)
962             {
963                 a59.Source =
964             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.thermos"
965                             a59Txt.Text = "Efficient Thermostat Set";
966             }
967
968             if (GetAchievementsData.insulateWaterCount >= 1)
969             {
970                 a60.Source =
971             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.waterTa"
972                             a60Txt.Text = "Insulated Water Tank";
973             }
974
975             if (GetAchievementsData.isolateHomeCount >= 1)
976             {
977                 a61.Source =
978             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.isolate"
979                             a61Txt.Text = "Insulated Home";
980             }
981
982             if (GetAchievementsData.ledLightBulbCount >= 1)
983             {
984                 a62.Source =
985             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.ledBulb"
986                             a62Txt.Text = "Led Lights Installed";
987             }
988
989             if (GetAchievementsData.fridgeCount >= 1)
990             {
991                 a63.Source =
992             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.turnDow"
993                             a63Txt.Text = "Turn Down The Fridge";
994             }
995
996             if (GetAchievementsData.draftSealCount >= 1)
997             {
998                 a64.Source =
999             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.sealDra"
999                             a64Txt.Text = "Drafts Sealed";
999             }
999
999             if (GetAchievementsData.ductSealCount >= 1)
999             {
999                 a65.Source =
999             ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.sealDuc
```

```
997         a65Txt.Text = "Ducts Sealed";
998     }
999
1000    if (GetAchievementsData.solarPanelCount >= 1)
1001    {
1002        a66.Source =
1003        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Energy.solar.p
1004            a66Txt.Text = "Installed Solar Panel";
1005        }
1006
1007    if (GetAchievementsData.reusableCount >= 1)
1008    {
1009        a67.Source =
1010        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Shopping.reWat
1011            a67Txt.Text = "Purchased Reusable Bottle";
1012        }
1013
1014    if (GetAchievementsData.reBatCount >= 1)
1015    {
1016        a68.Source =
1017        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Shopping.reBat
1018            a68Txt.Text = "Purchased Reusable Batteries";
1019        }
1020
1021    if (GetAchievementsData.cisternCount >= 1)
1022    {
1023        a69.Source =
1024        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.Cistern.
1025            a69Txt.Text = "Cistern Displacement System Installed";
1026        }
1027
1028    if (GetAchievementsData.rainBarrelCount >= 1)
1029    {
1030        a70.Source =
1031        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.rainBarr
1032            a70Txt.Text = "Rain Barrel Set Up";
1033        }
1034
1035    if (GetAchievementsData.wSShowerHeadCount >= 1)
1036    {
1037        a71.Source =
1038        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.Water.showerHe
1039            a71Txt.Text = "Water Saving Shower Head Installed";
1040        }
1041
1042    if (GetAchievementsData.fruitGardenCount >= 1)
1043    {
1044        a72.Source =
1045        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.OutOfdoors.fruit
1046            a72Txt.Text = "Fruit Garden Set Up";
1047        }
1048
1049    if (GetAchievementsData.herbGardenCount >= 1)
1050    {
1051        a73.Source =
1052        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.OutOfdoors.herb.
1053            a73Txt.Text = "Herb Garden Set Up";
1054        }
1055
1056    if (GetAchievementsData.vegetableGardenCount >= 1)
1057    {
1058        a74.Source =
1059        ImageSource.FromResource("Application_Green_Quake.Images.Achievements.OnceOff.OutOfdoors.veg.p
```

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

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <pages:PopupPage x:Class="Application_Green_Quake.Views.ProfilePage.BadgePopUp"
3             xmlns="http://xamarin.com/schemas/2014/forms"
4             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5             xmlns:pages="clr-
6 namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
7             xmlns:local="clr-namespace:Application_Green_Quake.Models"
8             BackgroundColor="#002a1e">
9
10    <Grid VerticalOptions="FillAndExpand" RowSpacing="0" BackgroundColor="#002a1e">
11        <Grid.RowDefinitions>
12            <RowDefinition Height="*"/>
13            <RowDefinition Height="2*"/>
14            <RowDefinition Height="*"/>
15        </Grid.RowDefinitions>
16
17        <StackLayout BackgroundColor="#002a1e"
18                     VerticalOptions="End">
19            <Label x:Name="badgeHeading" Text="Unranked" FontSize="25" TextColor="White"
20                   HorizontalTextAlignment="Center" />
21            <Label x:Name="badgeSubHeading" Text="" FontSize="15" TextColor="White"
22                   HorizontalTextAlignment="Center" />
23        </StackLayout>
24
25        <Image Grid.Row="1"
26               Aspect="AspectFill"
27               BackgroundColor="#002a1e"
28               Source="{local:ImageResource Application_Green_Quake.Images.lockTwo.png}"
29               x:Name="badge"/>
30
31        <StackLayout Grid.Row="2"
32                     BackgroundColor="#002a1e">
33            <Label x:Name="badgeTxt" Text="Next Rank: After 1 Action Log" FontSize="25"
34             TextColor="White" HorizontalTextAlignment="Center"/>
35        </StackLayout>
36
37    </Grid>
38 </pages:PopupPage>
```

```
1 /*! \class The BadgePopUp View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the BadgePopUp View Class. This class is the popup that appears when
8  * badge is tapped.
9  */
10
11
12 using Xamarin.Forms;
13 using Xamarin.Forms.Xaml;
14
15 namespace Application_Green_Quake.Views.ProfilePage
16 {
17     [XamlCompilation(XamlCompilationOptions.Compile)]
18     public partial class BadgePopUp
19     {
20         int num = 0;
21         int badgeState= 0;
22         /** The BadgePopUp Constructor
23          @param number is used to specify which badge it is
24          @param stage is used to specify what stage the badge is in
25         */
26         public BadgePopUp(int number, int stage)
27         {
28             num = number;
29             badgeState = stage;
30             InitializeComponent();
31             OnAppearing();
32         }
33         /** This function is called before the page is displayed. It displays the correct
34 information and badge based on what was passed into the
35         * constructor
36         */
37         protected override void OnAppearing()
38         {
39
40             if (num == 1 && badgeState == 1)
41             {
42                 badgeHeading.Text = "Habit Novice";
43                 badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
44                 badge.Source =
45                 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsOne.png");
46                 badgeTxt.Text = "Next Rank: After 5 Action Logs";
47             }
48             else if (num == 1 && badgeState == 2)
49             {
50                 badgeHeading.Text = "Habit Apprentice";
51                 badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
52                 badge.Source =
53                 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsTwo.png");
54                 badgeTxt.Text = "Next Rank: After 10 Action Logs";
55             }
56             else if (num == 1 && badgeState == 3)
57             {
58                 badgeHeading.Text = "Habit Adept";
59                 badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
60                 badge.Source =
61                 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsThree.png");
62                 badgeTxt.Text = "Next Rank: After 25 Action Logs";
63             }
64         }
65     }
66 }
```

```

57     else if (num == 1 && badgeState == 4)
58     {
59         badgeHeading.Text = "Habit Expert";
60         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
61         badge.Source =
62             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFour.png");
63         badgeTxt.Text = "Next Rank: After 50 Action Logs";
64     }
65     else if (num == 1 && badgeState == 5)
66     {
67         badgeHeading.Text = "Habit Master";
68         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
69         badge.Source =
70             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsFive.png");
71         badgeTxt.Text = "Next Rank: After 100 Action Logs";
72     }
73     else if (num == 1 && badgeState == 6)
74     {
75         badgeHeading.Text = "Habit Legend";
76         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
77         badge.Source =
78             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Habits.habitsSix.png");
79         badgeTxt.Text = "Max Rank Reached!";
80     }

81     if (num == 2 && badgeState == 1)
82     {
83         badgeHeading.Text = "Advanced Novice";
84         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
85         badge.Source =
86             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedOne.png");
87         badgeTxt.Text = "Next Rank: After 5 Action Logs";
88     }
89     else if (num == 2 && badgeState == 2)
90     {
91         badgeHeading.Text = "Advanced Apprentice";
92         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
93         badge.Source =
94             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedTwo.png");
95         badgeTxt.Text = "Next Rank: After 10 Action Logs";
96     }
97     else if (num == 2 && badgeState == 3)
98     {
99         badgeHeading.Text = "Advanced Habit Adept";
100        badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
101        badge.Source =
102            ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedThree.png");
103            badgeTxt.Text = "Next Rank: After 25 Action Logs";
104        }
105        else if (num == 2 && badgeState == 4)
106        {
107            badgeHeading.Text = "Advanced Expert";
108            badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
109            badge.Source =
110                ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFour.png");
111                badgeTxt.Text = "Next Rank: After 50 Action Logs";
112            }
113            else if (num == 2 && badgeState == 5)
114            {
115                badgeHeading.Text = "Advanced Master";
116                badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
117                badge.Source =
118                    ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedFive.png");

```

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

```

```
167         badgeHeading.Text = "Energy Novice";
168         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
169         badge.Source =
170     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyOne.png");
171         badgeTxt.Text = "Next Rank: After 5 Action Logs";
172     }
173     else if (num == 4 && badgeState == 2)
174     {
175         badgeHeading.Text = "Energy Apprentice";
176         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
177         badge.Source =
178     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyTwo.png");
179         badgeTxt.Text = "Next Rank: After 10 Action Logs";
180     }
181     else if (num == 4 && badgeState == 3)
182     {
183         badgeHeading.Text = "Energy Adept";
184         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
185         badge.Source =
186     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyThree.png");
187         badgeTxt.Text = "Next Rank: After 25 Action Logs";
188     }
189     else if (num == 4 && badgeState == 4)
190     {
191         badgeHeading.Text = "Energy Expert";
192         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
193         badge.Source =
194     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFour.png");
195         badgeTxt.Text = "Next Rank: After 50 Action Logs";
196     }
197     else if (num == 4 && badgeState == 5)
198     {
199         badgeHeading.Text = "Energy Master";
200         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
201         badge.Source =
202     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyFive.png");
203         badgeTxt.Text = "Next Rank: After 100 Action Logs";
204     }
205     else if (num == 4 && badgeState == 6)
206     {
207         badgeHeading.Text = "Energy Legend";
208         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
209         badge.Source =
210     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energySix.png");
211         badgeTxt.Text = "Max Rank Reached!";
212     }
213     if (num == 5 && badgeState == 1)
214     {
215         badgeHeading.Text = "Food And Drink Novice";
216         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
217         badge.Source =
218     ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDOne.png");
219         badgeTxt.Text = "Next Rank: After 5 Action Logs";
220     }
221     else if (num == 5 && badgeState == 2)
222     {
223         badgeHeading.Text = "Food And Drink Apprentice";
224         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
225         badge.Source =
226     ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDTwo.png");
227         badgeTxt.Text = "Next Rank: After 10 Action Logs";
228     }
```

```
222     else if (num == 5 && badgeState == 3)
223     {
224         badgeHeading.Text = "Food And Drink Adept";
225         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
226         badge.Source =
227             ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDThree.png");
228         badgeTxt.Text = "Next Rank: After 25 Action Logs";
229     }
230     else if (num == 5 && badgeState == 4)
231     {
232         badgeHeading.Text = "Food And Drink Expert";
233         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
234         badge.Source =
235             ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFour.png");
236         badgeTxt.Text = "Next Rank: After 50 Action Logs";
237     }
238     else if (num == 5 && badgeState == 5)
239     {
240         badgeHeading.Text = "Food And Drink Master";
241         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
242         badge.Source =
243             ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDFive.png");
244         badgeTxt.Text = "Next Rank: After 100 Action Logs";
245     }
246     else if (num == 5 && badgeState == 6)
247     {
248         badgeHeading.Text = "Food And Drink Legend";
249         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
250         badge.Source =
251             ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDSix.png");
252         badgeTxt.Text = "Max Rank Reached!";
253     }
254
255     if (num == 6 && badgeState == 1)
256     {
257         badgeHeading.Text = "Home Novice";
258         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
259         badge.Source =
260             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeOne.png");
261         badgeTxt.Text = "Next Rank: After 5 Action Logs";
262     }
263     else if (num == 6 && badgeState == 2)
264     {
265         badgeHeading.Text = "Home Apprentice";
266         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
267         badge.Source =
268             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeTwo.png");
269         badgeTxt.Text = "Next Rank: After 10 Action Logs";
270     }
271     else if (num == 6 && badgeState == 3)
272     {
273         badgeHeading.Text = "Home Adept";
274         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
275         badge.Source =
276             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeThree.png");
277         badgeTxt.Text = "Next Rank: After 25 Action Logs";
278     }
279     else if (num == 6 && badgeState == 4)
280     {
281         badgeHeading.Text = "Home Expert";
282         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
283         badge.Source =
```

```
    ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFour.png");
277        badgeTxt.Text = "Next Rank: After 50 Action Logs";
278    }
279    else if (num == 6 && badgeState == 5)
280    {
281        badgeHeading.Text = "Home Master";
282        badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
283        badge.Source =
284            ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFive.png");
285        badgeTxt.Text = "Next Rank: After 100 Action Logs";
286    }
287    else if (num == 6 && badgeState == 6)
288    {
289        badgeHeading.Text = "Home Legend";
290        badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
291        badge.Source =
292            ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeSix.png");
293        badgeTxt.Text = "Max Rank Reached!";
294    }
295    if (num == 7 && badgeState == 1)
296    {
297        badgeHeading.Text = "Outdoors Novice";
298        badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
299        badge.Source =
300            ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsOne.png");
301        badgeTxt.Text = "Next Rank: After 5 Action Logs";
302    }
303    else if (num == 7 && badgeState == 2)
304    {
305        badgeHeading.Text = "Outdoors Apprentice";
306        badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
307        badge.Source =
308            ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsTwo.png");
309        badgeTxt.Text = "Next Rank: After 10 Action Logs";
310    }
311    else if (num == 7 && badgeState == 3)
312    {
313        badgeHeading.Text = "Outdoors Adept";
314        badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
315        badge.Source =
316            ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsThree.png");
317        badgeTxt.Text = "Next Rank: After 25 Action Logs";
318    }
319    else if (num == 7 && badgeState == 4)
320    {
321        badgeHeading.Text = "Outdoors Expert";
322        badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
323        badge.Source =
324            ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFour.png");
325        badgeTxt.Text = "Next Rank: After 50 Action Logs";
326    }
327    else if (num == 7 && badgeState == 5)
328    {
329        badgeHeading.Text = "Outdoors Master";
330        badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
331        badge.Source =
332            ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFive.png");
333        badgeTxt.Text = "Next Rank: After 100 Action Logs";
334    }
335    else if (num == 7 && badgeState == 6)
```

```
331         badgeHeading.Text = "Outdoors Legend";
332         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
333         badge.Source =
334     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsSix.png");
335         badgeTxt.Text = "Max Rank Reached!";
336     }
337
338     if (num == 8 && badgeState == 1)
339     {
340         badgeHeading.Text = "Shopping Novice";
341         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
342         badge.Source =
343     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingOne.png");
344         badgeTxt.Text = "Next Rank: After 5 Action Logs";
345     }
346     else if (num == 8 && badgeState == 2)
347     {
348         badgeHeading.Text = "Shopping Apprentice";
349         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
350         badge.Source =
351     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingTwo.png");
352         badgeTxt.Text = "Next Rank: After 10 Action Logs";
353     }
354     else if (num == 8 && badgeState == 3)
355     {
356         badgeHeading.Text = "Shopping Adept";
357         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
358         badge.Source =
359     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingThree.png");
360         badgeTxt.Text = "Next Rank: After 25 Action Logs";
361     }
362     else if (num == 8 && badgeState == 4)
363     {
364         badgeHeading.Text = "Shopping Expert";
365         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
366         badge.Source =
367     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFour.png");
368         badgeTxt.Text = "Next Rank: After 50 Action Logs";
369     }
370     else if (num == 8 && badgeState == 5)
371     {
372         badgeHeading.Text = "Shopping Master";
373         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
374         badge.Source =
375     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFive.png");
376         badgeTxt.Text = "Next Rank: After 100 Action Logs";
377     }
378     else if (num == 8 && badgeState == 6)
379     {
380         badgeHeading.Text = "Shopping Legend";
381         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
382         badge.Source =
383     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingSix.png");
384         badgeTxt.Text = "Max Rank Reached!";
385     }
386
387     if (num == 9 && badgeState == 1)
388     {
389         badgeHeading.Text = "Travel Novice";
390         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
391         badge.Source =
392     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelOne.png");
393         badgeTxt.Text = "Next Rank: After 5 Action Logs";
394     }
```

```
386     }
387     else if (num == 9 && badgeState == 2)
388     {
389         badgeHeading.Text = "Travel Apprentice";
390         badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
391         badge.Source =
392             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelTwo.png");
393     }
394     else if (num == 9 && badgeState == 3)
395     {
396         badgeHeading.Text = "Travel Adept";
397         badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
398         badge.Source =
399             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelThree.png");
400         badgeTxt.Text = "Next Rank: After 25 Action Logs";
401     }
402     else if (num == 9 && badgeState == 4)
403     {
404         badgeHeading.Text = "Travel Expert";
405         badgeSubHeading.Text = "Badge awarded for loging 25 eco action!";
406         badge.Source =
407             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFour.png");
408         badgeTxt.Text = "Next Rank: After 50 Action Logs";
409     }
410     else if (num == 9 && badgeState == 5)
411     {
412         badgeHeading.Text = "Travel Master";
413         badgeSubHeading.Text = "Badge awarded for loging 50 eco action!";
414         badge.Source =
415             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFive.png");
416         badgeTxt.Text = "Next Rank: After 100 Action Logs";
417     }
418     else if (num == 9 && badgeState == 6)
419     {
420         badgeHeading.Text = "Travel Legend";
421         badgeSubHeading.Text = "Badge awarded for loging 100 eco action!";
422         badge.Source =
423             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelSix.png");
424         badgeTxt.Text = "Max Rank Reached!";
425     }
426
427     if (num == 10 && badgeState == 1)
428     {
429         badgeHeading.Text = "Waste Novice";
430         badgeSubHeading.Text = "Badge awarded for loging 1 eco action!";
431         badge.Source =
432             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteOne.png");
433         badgeTxt.Text = "Next Rank: After 5 Action Logs";
434     }
435     else if (num == 10 && badgeState == 2)
436     {
437         badgeHeading.Text = "Waste Apprentice";
438         badgeSubHeading.Text = "Badge awarded for loging 5 eco action!";
439         badge.Source =
440             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteTwo.png");
441         badgeTxt.Text = "Next Rank: After 10 Action Logs";
442     }
443     else if (num == 10 && badgeState == 3)
444     {
445         badgeHeading.Text = "Waste Adept";
446         badgeSubHeading.Text = "Badge awarded for loging 10 eco action!";
```

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

```
496         badgeHeading.Text = "Water Master";
497         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
498         badge.Source =
499         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFive.png");
500         badgeTxt.Text = "Next Rank: After 100 Action Logs";
501     }
502     else if (num == 11 && badgeState == 6)
503     {
504         badgeHeading.Text = "Water Legend";
505         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
506         badge.Source =
507         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterSix.png");
508         badgeTxt.Text = "Max Rank Reached!";
509     }
510
511     if (num == 12 && badgeState == 1)
512     {
513         badgeHeading.Text = "Work Novice";
514         badgeSubHeading.Text = "Badge awarded for logging 1 eco action!";
515         badge.Source =
516         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workOne.png");
517         badgeTxt.Text = "Next Rank: After 5 Action Logs";
518     }
519     else if (num == 12 && badgeState == 2)
520     {
521         badgeHeading.Text = "Work Apprentice";
522         badgeSubHeading.Text = "Badge awarded for logging 5 eco action!";
523         badge.Source =
524         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workTwo.png");
525         badgeTxt.Text = "Next Rank: After 10 Action Logs";
526     }
527     else if (num == 12 && badgeState == 3)
528     {
529         badgeHeading.Text = "Work Adept";
530         badgeSubHeading.Text = "Badge awarded for logging 10 eco action!";
531         badge.Source =
532         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workThree.png");
533         badgeTxt.Text = "Next Rank: After 25 Action Logs";
534     }
535     else if (num == 12 && badgeState == 4)
536     {
537         badgeHeading.Text = "Work Expert";
538         badgeSubHeading.Text = "Badge awarded for logging 25 eco action!";
539         badge.Source =
540         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFour.png");
541         badgeTxt.Text = "Next Rank: After 50 Action Logs";
542     }
543     else if (num == 12 && badgeState == 5)
544     {
545         badgeHeading.Text = "Work Master";
546         badgeSubHeading.Text = "Badge awarded for logging 50 eco action!";
547         badge.Source =
548         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFive.png");
549         badgeTxt.Text = "Next Rank: After 100 Action Logs";
550     }
551     else if (num == 12 && badgeState == 6)
552     {
553         badgeHeading.Text = "Work Legend";
554         badgeSubHeading.Text = "Badge awarded for logging 100 eco action!";
555         badge.Source =
556         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workSix.png");
557         badgeTxt.Text = "Max Rank Reached!";
558     }
559 }
```

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

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

```
53
54     <StackLayout Grid.Column="3" Grid.Row="0">
55         <StackLayout.GestureRecognizers>
56             <TapGestureRecognizer Tapped="NavigateToBadgePopUpFour"/>
57         </StackLayout.GestureRecognizers>
58         <Frame CornerRadius="60" HorizontalOptions="Start"
59             VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
60             <Image x:Name="a4" Source="{local:ImageResource
61                 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
62         </Frame>
63
64         <Label x:Name="a4Txt" Text="Locked" TextColor="White"
65             HorizontalTextAlignment="Center" />
66     </StackLayout>
67
68     <StackLayout Grid.Column="0" Grid.Row="1">
69         <StackLayout.GestureRecognizers>
70             <TapGestureRecognizer Tapped="NavigateToBadgePopUpFive"/>
71         </StackLayout.GestureRecognizers>
72         <Frame CornerRadius="60" HorizontalOptions="Start"
73             VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
74             <Image x:Name="a5" Source="{local:ImageResource
75                 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
76         </Frame>
77
78         <Label x:Name="a5Txt" Text="Locked " TextColor="White"
79             HorizontalTextAlignment="Center"/>
80     </StackLayout>
81
82     <StackLayout Grid.Column="1" Grid.Row="1">
83         <StackLayout.GestureRecognizers>
84             <TapGestureRecognizer Tapped="NavigateToBadgePopUpSix"/>
85         </StackLayout.GestureRecognizers>
86         <Frame CornerRadius="60" HorizontalOptions="Start"
87             VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
88             <Image x:Name="a6" Source="{local:ImageResource
89                 Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
90         </Frame>
91
92         <Label x:Name="a6Txt" Text="Locked" TextColor="White"
93             HorizontalTextAlignment="Center"/>
94     </StackLayout>
95
96
97     <StackLayout Grid.Column="2" Grid.Row="1">
98         <StackLayout.GestureRecognizers>
99             <TapGestureRecognizer Tapped="NavigateToBadgePopUpSeven"/>
100        </StackLayout.GestureRecognizers>
101        <Frame CornerRadius="60" HorizontalOptions="Start"
102            VerticalOptions="Start" Margin="0" Padding="0" BackgroundColor="#33554b">
103            <Image x:Name="a7" Source="{local:ImageResource
104                Application_Green_Quake.Images.lockTwo.png}" HorizontalOptions="CenterAndExpand"/>
105        </Frame>
106
107        <Label x:Name="a7Txt" Text="Locked" TextColor="White"
108            HorizontalTextAlignment="Center" />
109    </StackLayout>
110
111    <StackLayout Grid.Column="3" Grid.Row="1">
112        <StackLayout.GestureRecognizers>
113            <TapGestureRecognizer Tapped="NavigateToBadgePopUpEight"/>
114        </StackLayout.GestureRecognizers>
```

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

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

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

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

```
112     ImageSource.FromResource("Application_Green_Quake.Images.Badges.Advanced.advancedSix.png");
113         a2Txt.Text = "Advanced Legend";
114         advancedStage = 6;
115     }
116
117     if (GetBadgeData.communityLog > 0 && GetBadgeData.communityLog < 5)
118     {
119         a3.Source =
120             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityOne.png");
121             a3Txt.Text = "Community Novice";
122             communityStage = 1;
123     }
124     else if (GetBadgeData.communityLog >= 5 && GetBadgeData.communityLog < 10)
125     {
126         a3.Source =
127             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityTwo.png");
128             a3Txt.Text = "Community Apprentice";
129             communityStage = 2;
130     }
131     else if (GetBadgeData.communityLog >= 10 && GetBadgeData.communityLog < 25)
132     {
133         a3.Source =
134             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityThree.png");
135             a3Txt.Text = "Community Adept";
136             communityStage = 3;
137     }
138     else if (GetBadgeData.communityLog >= 25 && GetBadgeData.communityLog < 50)
139     {
140         a3.Source =
141             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFour.png");
142             a3Txt.Text = "Community Expert";
143             communityStage = 4;
144     }
145     else if (GetBadgeData.communityLog >= 50 && GetBadgeData.communityLog < 100)
146     {
147         a3.Source =
148             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Community.communityFive.png");
149             a3Txt.Text = "Community Master";
150             communityStage = 5;
151     }
152
153     if (GetBadgeData.energyLog > 0 && GetBadgeData.energyLog < 5)
154     {
155         a4.Source =
156             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyOne.png");
157             a4Txt.Text = "Energy Novice";
158             energyStage = 1;
159     }
160     else if (GetBadgeData.energyLog >= 5 && GetBadgeData.energyLog < 10)
161     {
162         a4.Source =
163             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Energy.energyTwo.png");
164             a4Txt.Text = "Energy Apprentice";
165             energyStage = 2;
166     }
167     else if (GetBadgeData.energyLog >= 10 && GetBadgeData.energyLog < 25)
```

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

```

220     else if (GetBadgeData.foodDrinkLog >= 100)
221     {
222         a5.Source =
223             ImageSource.FromResource("Application_Green_Quake.Images.Badges.FD.fDSix.png");
224         a5Txt.Text = "F & D Legend";
225         foodDrinkStage = 6;
226     }
227
228     if (GetBadgeData.homeLog > 0 && GetBadgeData.homeLog < 5)
229     {
230         a6.Source =
231             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeOne.png");
232         a6Txt.Text = "Home Novice";
233         homeStage = 1;
234     }
235     else if (GetBadgeData.homeLog >= 5 && GetBadgeData.homeLog < 10)
236     {
237         a6.Source =
238             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeTwo.png");
239         a6Txt.Text = "Home Apprentice";
240         homeStage = 2;
241     }
242     else if (GetBadgeData.homeLog >= 10 && GetBadgeData.homeLog < 25)
243     {
244         a6.Source =
245             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeThree.png");
246         a6Txt.Text = "Home Adept";
247         homeStage = 3;
248     }
249     else if (GetBadgeData.homeLog >= 25 && GetBadgeData.homeLog < 50)
250     {
251         a6.Source =
252             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFour.png");
253         a6Txt.Text = "Home Expert";
254         homeStage = 4;
255     }
256     else if (GetBadgeData.homeLog >= 50 && GetBadgeData.homeLog < 100)
257     {
258         a6.Source =
259             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeFive.png");
260         a6Txt.Text = "Home Master";
261         homeStage = 5;
262     }
263     else if (GetBadgeData.homeLog >= 100)
264     {
265         a6.Source =
266             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Home.homeSix.png");
267         a6Txt.Text = "Home Legend";
268         homeStage = 6;
269     }
270
271     if (GetBadgeData.outdoorsLog > 0 && GetBadgeData.outdoorsLog < 5)
272     {
273         a7.Source =
274             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsOne.png");
275         a7Txt.Text = "Outdoors Novice";
276         outdoorsStage = 1;
277     }
278     else if (GetBadgeData.outdoorsLog >= 5 && GetBadgeData.outdoorsLog < 10)
279     {
280         a7.Source =
281             ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsTwo.png");
282         a7Txt.Text = "Outdoors Apprentice";

```

```
274         outdoorsStage = 2;
275     }
276     else if (GetBadgeData.outdoorsLog >= 10 && GetBadgeData.outdoorsLog < 25)
277     {
278         a7.Source = ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsThree.png");
279         a7Txt.Text = "Outdoors Adept";
280         outdoorsStage = 3;
281     }
282     else if (GetBadgeData.outdoorsLog >= 25 && GetBadgeData.outdoorsLog < 50)
283     {
284         a7.Source =
285 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFour.png");
286         a7Txt.Text = "Outdoors Expert";
287         outdoorsStage = 4;
288     }
289     else if (GetBadgeData.outdoorsLog >= 50 && GetBadgeData.outdoorsLog < 100)
290     {
291         a7.Source =
292 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsFive.png");
293         a7Txt.Text = "Outdoors Master";
294         outdoorsStage = 5;
295     }
296     else if (GetBadgeData.outdoorsLog >= 100)
297     {
298         a7.Source =
299 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Outdoors.outdoorsSix.png");
300         a7Txt.Text = "Outdoors Legend";
301         outdoorsStage = 6;
302     }

303     if (GetBadgeData.shoppingLog > 0 && GetBadgeData.shoppingLog < 5)
304     {
305         a8.Source =
306 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingOne.png");
307         a8Txt.Text = "Shopping Novice";
308         shoppingStage = 1;
309     }
310     else if (GetBadgeData.shoppingLog >= 5 && GetBadgeData.shoppingLog < 10)
311     {
312         a8.Source =
313 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingTwo.png");
314         a8Txt.Text = "Shopping Apprentice";
315         shoppingStage = 2;
316     }
317     else if (GetBadgeData.shoppingLog >= 10 && GetBadgeData.shoppingLog < 25)
318     {
319         a8.Source =
320 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingThree.png");
321         a8Txt.Text = "Shopping Adept";
322         shoppingStage = 3;
323     }
324     else if (GetBadgeData.shoppingLog >= 25 && GetBadgeData.shoppingLog < 50)
325     {
326         a8.Source =
327 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFour.png");
328         a8Txt.Text = "Shopping Expert";
329         shoppingStage = 4;
330     }
331     else if (GetBadgeData.shoppingLog >= 50 && GetBadgeData.shoppingLog < 100)
332     {
333         a8.Source =
334 ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingFive.png");
```

```
328         a8Txt.Text = "Shopping Master";
329         shoppingStage = 5;
330     }
331     else if (GetBadgeData.shoppingLog >= 100)
332     {
333         a8.Source =
334         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Shopping.shoppingSix.png");
335         a8Txt.Text = "Shopping Legend";
336         shoppingStage = 6;
337     }
338
339     if (GetBadgeData.travelLog > 0 && GetBadgeData.travelLog < 5)
340     {
341         a9.Source =
342         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelOne.png");
343         a9Txt.Text = "Travel Novice";
344         travelStage = 1;
345     }
346     else if (GetBadgeData.travelLog >= 5 && GetBadgeData.travelLog < 10)
347     {
348         a9.Source =
349         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelTwo.png");
350         a9Txt.Text = "Travel Apprentice";
351         travelStage = 2;
352     }
353     else if (GetBadgeData.travelLog >= 10 && GetBadgeData.travelLog < 25)
354     {
355         a9.Source =
356         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelThree.png");
357         a9Txt.Text = "Travel Adept";
358         travelStage = 3;
359     }
360     else if (GetBadgeData.travelLog >= 25 && GetBadgeData.travelLog < 50)
361     {
362         a9.Source =
363         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFour.png");
364         a9Txt.Text = "Travel Expert";
365         travelStage = 4;
366     }
367     else if (GetBadgeData.travelLog >= 50 && GetBadgeData.travelLog < 100)
368     {
369         a9.Source =
370         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelFive.png");
371         a9Txt.Text = "Travel Master";
372         travelStage = 5;
373     }
374     else if (GetBadgeData.travelLog >= 100)
375     {
376         a9.Source =
377         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Travel.travelSix.png");
378         a9Txt.Text = "Travel Legend";
379         travelStage = 6;
380     }
381
382     if (GetBadgeData.wasteLog > 0 && GetBadgeData.wasteLog < 5)
383     {
384         a10.Source =
385         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteOne.png");
386         a10Txt.Text = "Waste Novice";
387         wasteStage = 1;
388     }
389     else if (GetBadgeData.wasteLog >= 5 && GetBadgeData.wasteLog < 10)
390     {
391         a10.Source =
392         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteTwo.png");
393         a10Txt.Text = "Waste Apprentice";
394         wasteStage = 2;
395     }
396     else if (GetBadgeData.wasteLog >= 10 && GetBadgeData.wasteLog < 25)
397     {
398         a10.Source =
399         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteThree.png");
400         a10Txt.Text = "Waste Expert";
401         wasteStage = 3;
402     }
403     else if (GetBadgeData.wasteLog >= 25 && GetBadgeData.wasteLog < 50)
404     {
405         a10.Source =
406         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFour.png");
407         a10Txt.Text = "Waste Master";
408         wasteStage = 4;
409     }
410     else if (GetBadgeData.wasteLog >= 50 && GetBadgeData.wasteLog < 100)
411     {
412         a10.Source =
413         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFive.png");
414         a10Txt.Text = "Waste Legend";
415         wasteStage = 5;
416     }
417 }
```

```
383         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteTwo.png");
384         a10Txt.Text = "Waste Apprentice";
385         wasteStage = 2;
386     }
387     else if (GetBadgeData.wasteLog >= 10 && GetBadgeData.wasteLog < 25)
388     {
389         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteThree.png");
390         a10Txt.Text = "Waste Adept";
391         wasteStage = 3;
392     }
393     else if (GetBadgeData.wasteLog >= 25 && GetBadgeData.wasteLog < 50)
394     {
395         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFour.png");
396         a10Txt.Text = "Waste Expert";
397         wasteStage = 4;
398     }
399     else if (GetBadgeData.wasteLog >= 50 && GetBadgeData.wasteLog < 100)
400     {
401         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteFive.png");
402         a10Txt.Text = "Waste Master";
403         wasteStage = 5;
404     }
405     else if (GetBadgeData.wasteLog >= 100)
406     {
407         a10.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Waste.wasteSix.png");
408         a10Txt.Text = "Waste Legend";
409         wasteStage = 6;
410     }

411
412     if (GetBadgeData.waterLog > 0 && GetBadgeData.waterLog < 5)
413     {
414         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterOne.png");
415         a11Txt.Text = "Water Novice";
416         waterStage = 1;
417     }
418     else if (GetBadgeData.waterLog >= 5 && GetBadgeData.waterLog < 10)
419     {
420         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterTwo.png");
421         a11Txt.Text = "Water Apprentice";
422         waterStage = 2;
423     }
424     else if (GetBadgeData.waterLog >= 10 && GetBadgeData.waterLog < 25)
425     {
426         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterThree.png");
427         a11Txt.Text = "Water Adept";
428         waterStage = 3;
429     }
430     else if (GetBadgeData.waterLog >= 25 && GetBadgeData.waterLog < 50)
431     {
432         a11.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFour.png");
433         a11Txt.Text = "Water Expert";
434         waterStage = 4;
435     }
436     else if (GetBadgeData.waterLog >= 50 && GetBadgeData.waterLog < 100)
```

```
437     {
438         a11.Source =
439         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterFive.png");
440         a11Txt.Text = "Water Master";
441         waterStage = 5;
442     }
443     else if (GetBadgeData.waterLog >= 100)
444     {
445         a11.Source =
446         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Water.waterSix.png");
447         a11Txt.Text = "Water Legend";
448         waterStage = 6;
449     }
450
451     if (GetBadgeData.workLog > 0 && GetBadgeData.workLog < 5)
452     {
453         a12.Source =
454         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workOne.png");
455         a12Txt.Text = "Work Novice";
456         workStage = 1;
457     }
458     else if (GetBadgeData.workLog >= 5 && GetBadgeData.workLog < 10)
459     {
460         a12.Source =
461         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workTwo.png");
462         a12Txt.Text = "Work Apprentice";
463         workStage = 2;
464     }
465     else if (GetBadgeData.workLog >= 10 && GetBadgeData.workLog < 25)
466     {
467         a12.Source =
468         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workThree.png");
469         a12Txt.Text = "Work Adept";
470         workStage = 3;
471     }
472     else if (GetBadgeData.workLog >= 25 && GetBadgeData.workLog < 50)
473     {
474         a12.Source =
475         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFour.png");
476         a12Txt.Text = "Work Expert";
477         workStage = 4;
478     }
479     else if (GetBadgeData.workLog >= 50 && GetBadgeData.workLog < 100)
480     {
481         a12.Source =
482         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workFive.png");
483         a12Txt.Text = "Work Master";
484         workStage = 5;
485     }
486     else if (GetBadgeData.workLog >= 100)
487     {
488         a12.Source =
489         ImageSource.FromResource("Application_Green_Quake.Images.Badges.Work.workSix.png");
490         a12Txt.Text = "Work Legend";
491         workStage = 6;
492     }
493 }
494 */
495 private async void NavigateToBadgePopUpOne(object sender, EventArgs e)
496 {
497     int number = 1;
498     await PopupNavigation.Instance.PushAsync(new BadgePopUp(number, habitsStage));
499 }
```

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

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

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

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

```
1 /*! \class The TopTabProfile View Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the TopTabProfile View Class. The profile page class.
8  */
9 using Application_Green_Quake.Models;
10 using Application_Green_Quake.ViewModels;
11 using Firebase.Database;
12 using Firebase.Database.Query;
13 using Firebase.Storage;
14 using Rg.Plugins.Popup.Services;
15 using System;
16 using Xamarin.Forms;
17 using Xamarin.Forms.Xaml;
18
19 namespace Application_Green_Quake.Views.ProfilePage
20 {
21     [XamlCompilation(XamlCompilationOptions.Compile)]
22     public partial class TopTabProfile : ContentPage
23     {
24         IAuth auth;
25         float progress = 0;
26         string bioInput = "";
27         float count = 0;
28
29         public TopTabProfile()
30         {
31             InitializeComponent();
32             auth = DependencyService.Get<IAuth>();
33             OnAppearing();
34         }
35         /** This function is called before the page is displayed. It displays the images as
36          the criteria are met and also the bio and username. It also
37          * calculates the players level and progress and displays the progress to the next
38          level on a progress bar.
39          */
40         protected override async void OnAppearing()
41         {
42             Username.Text = GetData.username;
43             try
44             {
45                 ProfilePic.Source = await new FirebaseStorage("application-green-
46                 quake.appspot.com")
47                     .Child(auth.GetUid())
48                     .Child("Profile.jpg")
49                     .GetDownloadUrlAsync();
50             }
51             catch (Exception e)
52             {
53                 Console.WriteLine(e);
54             }
55
56             // Calculate the progress for the next level
57             progress = (float)GetData.points / 10;
58             progress = (int)((decimal)progress % 1) * 10;
59
60             progress = progress / 10;
```

```
58
59         count = progress;
60         count = count * 10;
61
62         // Set the theLvl and animate the progress bar
63         theLevel.Text = "Lvl: " + GetData.lvl.ToString() + " Points: " +
64         count.ToString() + " /10";
65         await progressbar.ProgressTo(progress, 500, Easing.Linear);
66
67         if (GetData.lvl == 1)
68         {
69             mosiac.Source =
70             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineOne.jpg");
71         }
72         else if (GetData.lvl == 2)
73         {
74             mosiac.Source =
75             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineTwo.jpg");
76         }
77         else if (GetData.lvl == 3)
78         {
79             mosiac.Source =
80             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineThree.jpg");
81         }
82         else if (GetData.lvl == 4)
83         {
84             mosiac.Source =
85             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1outlineFour.jpg");
86         }
87         else if (GetData.lvl == 5)
88         {
89             mosiac.Source =
90             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m1.jpg");
91         }
92         else if (GetData.lvl == 6)
93         {
94             mosiac.Source =
95             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineOne.jpg");
96         }
97         else if (GetData.lvl == 7)
98         {
99             mosiac.Source =
100            ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineTwo.jpg");
101        }
102        else if (GetData.lvl == 8)
103        {
104            mosiac.Source =
105            ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineThree.jpg");
106        }
107        else if (GetData.lvl == 9)
108        {
109            mosiac.Source =
110            ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2outlineFour.jpg");
111        }
112        else if (GetData.lvl == 10)
113        {
114            mosiac.Source =
115            ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m2.jpg");
116        }
117        else if (GetData.lvl == 11)
118        {
119            mosiac.Source =
120            ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineOne.jpg");
121        }
122    }
123}
```

```
109     }
110     else if (GetData.lv1 == 12)
111     {
112         mosiac.Source =
113             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineTwo.jpg");
114         }
115         else if (GetData.lv1 == 13)
116         {
117             mosiac.Source =
118                 ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineThree.jpg");
119                 }
120                 else if (GetData.lv1 == 14)
121                 {
122                     mosiac.Source =
123                         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3outlineFour.jpg");
124                         }
125                         else if (GetData.lv1 == 15)
126                         {
127                             mosiac.Source =
128                                 ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m3.jpg");
129                                 }
130                                 else if (GetData.lv1 == 16)
131                                 {
132                                     mosiac.Source =
133                                         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineOne.jpg");
134                                         }
135                                         else if (GetData.lv1 == 17)
136                                         {
137                                             mosiac.Source =
138                                                 ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineTwo.jpg");
139                                                 }
140                                                 else if (GetData.lv1 == 18)
141                                                 {
142                                                     mosiac.Source =
143                                                         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineThree.jpg");
144                                                         }
145                                                         else if (GetData.lv1 == 19)
146                                                         {
147                                                             mosiac.Source =
148                                                               ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4outlineFour.jpg");
149                                                               }
150                                                               else if (GetData.lv1 == 20)
151                                                               {
152                                                                   mosiac.Source =
153                         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m4.jpg");
154                         }
155                         else if (GetData.lv1 == 21)
156                         {
157                             mosiac.Source =
158                               ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineOne.jpg");
159                               }
```

```
160         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5outlineFour.jpg");
161     }
162     else if (GetData.lv1 == 25)
163     {
164         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m5.jpg");
165     }
166     else if (GetData.lv1 == 26)
167     {
168         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineOne.jpg");
169     }
170     else if (GetData.lv1 == 27)
171     {
172         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineTwo.jpg");
173     }
174     else if (GetData.lv1 == 28)
175     {
176         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineThree.jpg");
177     }
178     else if (GetData.lv1 == 29)
179     {
180         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6outlineFour.jpg");
181     }
182     else if (GetData.lv1 == 30)
183     {
184         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m6.jpg");
185     }
186     else if (GetData.lv1 == 31)
187     {
188         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineOne.jpg");
189     }
190     else if (GetData.lv1 == 32)
191     {
192         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineTwo.jpg");
193     }
194     else if (GetData.lv1 == 33)
195     {
196         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineThree.jpg");
197     }
198     else if (GetData.lv1 == 34)
199     {
200         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7outlineFour.jpg");
201     }
202     else if (GetData.lv1 == 35)
203     {
204         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m7.jpg");
205     }
206     else if (GetData.lv1 == 36)
207     {
208         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineOne.jpg");
209     }
```

```
210     else if (GetData.lv1 == 37)
211     {
212         mosiac.Source =
213         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineTwo.jpg");
214     }
215     else if (GetData.lv1 == 38)
216     {
217         mosiac.Source =
218         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineThree.jpg");
219     }
220     else if (GetData.lv1 == 39)
221     {
222         mosiac.Source =
223         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8outlineFour.jpg");
224     }
225     else if (GetData.lv1 == 40)
226     {
227         mosiac.Source =
228         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m8.jpg");
229     }
230     else if (GetData.lv1 == 41)
231     {
232         mosiac.Source =
233         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineOne.jpg");
234     }
235     else if (GetData.lv1 == 42)
236     {
237         mosiac.Source =
238         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineTwo.jpg");
239     }
240     else if (GetData.lv1 == 43)
241     {
242         mosiac.Source =
243         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineThree.jpg");
244     }
245     else if (GetData.lv1 == 44)
246     {
247         mosiac.Source =
248         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m9outlineFour.jpg");
249     }
250     else if (GetData.lv1 == 45)
251     {
252         mosiac.Source =
253         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineOne.jpg");
254     }
255     else if (GetData.lv1 == 46)
256     {
257         mosiac.Source =
258         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineTwo.jpg");
259     }
260     else if (GetData.lv1 == 47)
261     {
262         mosiac.Source =
263         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineThree.jpg");
264     }
265     else if (GetData.lv1 == 48)
266     {
267         mosiac.Source =
268         ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineFour.jpg");
269     }
270     else if (GetData.lv1 == 49)
271     {
272         mosiac.Source =
```

```
261     }
262     else if (GetData.lv1 == 50)
263     {
264         mosiac.Source =
265     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10outlineFour.jpg");
266     }
267     else if (GetData.lv1 == 51)
268     {
269         mosiac.Source =
270     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m10.jpg");
271     }
272     else if (GetData.lv1 == 52)
273     {
274         mosiac.Source =
275     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineOne.jpg");
276     }
277     else if (GetData.lv1 == 53)
278     {
279         mosiac.Source =
280     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineTwo.jpg");
281     }
282     else if (GetData.lv1 == 54)
283     {
284         mosiac.Source =
285     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11outlineFour.jpg");
286     }
287     else if (GetData.lv1 == 55)
288     {
289         mosiac.Source =
290     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m11.jpg");
291     }
292     else if (GetData.lv1 == 56)
293     {
294         mosiac.Source =
295     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineOne.jpg");
296     }
297     else if (GetData.lv1 == 57)
298     {
299         mosiac.Source =
300     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineTwo.jpg");
301     }
302     else if (GetData.lv1 == 58)
303     {
304         mosiac.Source =
305     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineThree.jpg");
306     }
307     else if (GetData.lv1 == 59)
308     {
309         mosiac.Source =
310     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12outlineFour.jpg");
311     }
312     else if (GetData.lv1 == 60)
313     {
314         mosiac.Source =
315     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m12.jpg");
316     }
317     else if (GetData.lv1 == 61)
318     {
319         mosiac.Source =
320     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineOne.jpg");
321     }
322     else if (GetData.lv1 == 62)
```

```
311     {
312         mosiac.Source =
313     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineTwo.jpg");
314     }
315     else if (GetData.lv1 == 63)
316     {
317         mosiac.Source =
318     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineThree.jpg");
319     }
320     else if (GetData.lv1 == 64)
321     {
322         mosiac.Source =
323     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13outlineFour.jpg");
324     }
325     else if (GetData.lv1 == 65)
326     {
327         mosiac.Source =
328     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m13.jpg");
329     }
330     else if (GetData.lv1 == 66)
331     {
332         mosiac.Source =
333     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineOne.jpg");
334     }
335     else if (GetData.lv1 == 67)
336     {
337         mosiac.Source =
338     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineTwo.jpg");
339     }
340     else if (GetData.lv1 == 68)
341     {
342         mosiac.Source =
343     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineThree.jpg");
344     }
345     else if (GetData.lv1 == 69)
346     {
347         mosiac.Source =
348     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14outlineFour.jpg");
349     }
350     else if (GetData.lv1 == 70)
351     {
352         mosiac.Source =
353     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m14.jpg");
354     }
355     else if (GetData.lv1 == 71)
356     {
357         mosiac.Source =
358     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineOne.jpg");
359     }
360     else if (GetData.lv1 == 72)
361     {
362         mosiac.Source =
363     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineTwo.jpg");
364     }
365     else if (GetData.lv1 == 73)
366     {
367         mosiac.Source =
368     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineThree.jpg");
369     }
370     else if (GetData.lv1 == 74)
371     {
372         mosiac.Source =
373     ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15outlineFour.jpg");
```

```
361     }
362     else if (GetData_LVL == 75)
363     {
364         mosiac.Source =
365             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m15.jpg");
366     }
367     else if (GetData_LVL == 76)
368     {
369         mosiac.Source =
370             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineOne.jpg");
371     }
372     else if (GetData_LVL == 77)
373     {
374         mosiac.Source =
375             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineTwo.jpg");
376     }
377     else if (GetData_LVL == 78)
378     {
379         mosiac.Source =
380             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineThree.jpg");
381     }
382     else if (GetData_LVL == 79)
383     {
384         mosiac.Source =
385             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16outlineFour.jpg");
386     }
387     else if (GetData_LVL == 80)
388     {
389         mosiac.Source =
390             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m16.jpg");
391     }
392     else if (GetData_LVL == 81)
393     {
394         mosiac.Source =
395             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineOne.jpg");
396     }
397     else if (GetData_LVL == 82)
398     {
399         mosiac.Source =
400             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineTwo.jpg");
401     }
402     else if (GetData_LVL == 83)
403     {
404         mosiac.Source =
405             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineThree.jpg");
406     }
407     else if (GetData_LVL == 84)
408     {
409         mosiac.Source =
410             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17outlineFour.jpg");
411     }
412     else if (GetData_LVL == 85)
413     {
414         mosiac.Source =
415             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m17.jpg");
416     }
417     else if (GetData_LVL == 86)
418     {
419         mosiac.Source =
420             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineOne.jpg");
421     }
422     else if (GetData_LVL == 87)
423     {
424 }
```

```
412         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineTwo.jpg");
413     }
414     else if (GetData.lv1 == 88)
415     {
416         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineThree.jpg");
417     }
418     else if (GetData.lv1 == 89)
419     {
420         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18outlineFour.jpg");
421     }
422     else if (GetData.lv1 == 90)
423     {
424         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m18.jpg");
425     }
426     else if (GetData.lv1 == 91)
427     {
428         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineOne.jpg");
429     }
430     else if (GetData.lv1 == 92)
431     {
432         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineTwo.jpg");
433     }
434     else if (GetData.lv1 == 93)
435     {
436         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineThree.jpg");
437     }
438     else if (GetData.lv1 == 94)
439     {
440         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19outlineFour.jpg");
441     }
442     else if (GetData.lv1 == 95)
443     {
444         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m19.jpg");
445     }
446     else if (GetData.lv1 == 96)
447     {
448         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineOne.jpg");
449     }
450     else if (GetData.lv1 == 97)
451     {
452         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineTwo.jpg");
453     }
454     else if (GetData.lv1 == 98)
455     {
456         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineThree.jpg");
457     }
458     else if (GetData.lv1 == 99)
459     {
460         mosiac.Source =
ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20outlineFour.jpg");
461     }
```

```
462     else if (GetData.lv1 == 100)
463     {
464         mosiac.Source =
465             ImageSource.FromResource("Application_Green_Quake.Images.Mosaics.m20.jpg");
466     }
467
468     GetData data = new GetData();
469     data.SetLv1();
470 }
471 /**
472 private async void ImageClicked(object sender, EventArgs e)
473 {
474     await PopupNavigation.Instance.PushAsync(new UploadImagePopUp());
475 }
476 /**
477 private async void NavigateToTrophyCase(object sender, EventArgs e)
478 {
479     await Navigation.PushAsync(new TrophyCase(0));
480 }
481 /**
482 private async void NavigateToAchievements(object sender, EventArgs e)
483 {
484     await Navigation.PushAsync(new TrophyCase(1));
485 }
486 /**
487 private async void NavigateToBadges(object sender, EventArgs e)
488 {
489     await Navigation.PushAsync(new TrophyCase(2));
490 }
491 /**
492 private async void SaveText(object sender, EventArgs e)
493 {
494     FirebaseClient firebaseClient = new FirebaseClient("https://application-green-
495     quake-default-rtdb.firebaseio.com/");
496
497     bioInput = Bio.Text;
498
499     await firebaseClient
500         .Child("users")
501             .Child(auth.GetUserId())
502                 .PutAsync(new Users() {username = GetData.username ,bio = bioInput,
503 nation = GetData.nation});
504             }
505     }
506 }
507 }
508 }
```

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:views="clr-
4 namespace:Application_Green_Quake.Views" xmlns:views1="clr-
5 namespace:Application_Green_Quake.Views.ProfilePage"
6             xmlns:models="clr-namespace:Application_Green_Quake.Models;assembly=Application
7 Green Quake"
8             x:Class="Application_Green_Quake.Views.ProfilePage.TrophyCase"
9             NavigationPage.HasBackButton="False"
10            NavigationPage.HasNavigationBar="False">
11 <ContentPage Title="Trophies">
12     <ContentPage.Content>
13         <ScrollView>
14             <Grid HorizontalOptions="Fill" VerticalOptions="Fill" Padding="10,30,10,10"
15 Row="20">
16                 <Grid.RowDefinitions>
17                     <RowDefinition Height="*"/>
18                     <RowDefinition Height="*"/>
19                     <RowDefinition Height="*"/>
20                     <RowDefinition Height="*"/>
21                 </Grid.RowDefinitions>
22                 <Grid.ColumnDefinitions>
23                     <ColumnDefinition Width="*"/>
24                 </Grid.ColumnDefinitions>
25                 <StackLayout Grid.Row="0" >
26                     <Image x:Name="t1"
27                           Source="{models:ImageResource
28 Application_Green_Quake.Images.lockOne.png}"
29                           HorizontalOptions="CenterAndExpand">
30                         </Image>
31                         <Label x:Name="t1Txt" Text="1000 Points to unlock" TextColor="Black"
32 HorizontalTextAlignment="Center"/>
33                     </StackLayout>
34                     <StackLayout Grid.Row="1">
35                         <Image x:Name="t2"
36                           Source="{models:ImageResource
37 Application_Green_Quake.Images.lockOne.png}"
38                           HorizontalOptions="CenterAndExpand">
39                         </Image>
40                         <Label x:Name="t2Txt" Text="500 Points to unlock" TextColor="Black"
41 HorizontalTextAlignment="Center"/>
42                     </StackLayout>
43                     <StackLayout Grid.Row="2">
44                         <Image x:Name="t3"
45                           Source="{models:ImageResource
46 Application_Green_Quake.Images.lockOne.png}"
47                           HorizontalOptions="CenterAndExpand">
48                         </Image>
49                         <Label x:Name="t3Txt" Text="250 Points to unlock" TextColor="Black"
50 HorizontalTextAlignment="Center"/>
51                     </StackLayout>
52                 </Grid>
53             </ScrollView>
54         </ContentPage.Content>
```

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

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

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <pages:PopupPage x:Class="Application_Green_Quake.Views.ProfilePage.UploadImagePopUp"
3             xmlns="http://xamarin.com/schemas/2014/forms"
4             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5             xmlns:pages="clr-
6 namespace:Rg.Plugins.Popup.Pages;assembly=Rg.Plugins.Popup"
7             BackgroundColor="##002a1e">
8
9     <StackLayout HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" >
10         <Image x:Name="ChosenImage" HeightRequest="200"/></Image>
11         <Button x:Name="Take" TextColor="White" BackgroundColor="#50C878" Text="Take Photo"
12             Clicked="CapturePhotoClicked"></Button>
13         <Button x:Name="Pick" TextColor="White" BackgroundColor="#50C878" Text="Pick From
14             Storage" Clicked="UplaodPhotoClicked"></Button>
15         <Button x:Name="Save" TextColor="White" BackgroundColor="#50C878" Text="Save"
16             Clicked="storeImageClicked"></Button>
17     </StackLayout>
18
19 </pages:PopupPage>
```

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

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

```
119 }  
120 }
```

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

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

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

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

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

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

```
1 /*! \class The AuthDroid Native Android Class
2  * \author Peter Lucan, 4th Year Software Development student at IT Carlow, C00228946,
3  * c00228956@itcarlow.ie
4  * \date 28/04/2021
5  * \section desc_sec Description
6  *
7  * Description: This is the AuthDroid Native Android Class. This is implements native
8  * functions to check if a user is signed in, or try and log in, to try and
9  * sign up to get their uid and to get a password rest email.
10 *
11 */
12
13 using System;
14 using System.Threading.Tasks;
15 using Xamarin.Forms;
16 using Application_Green_Quake.Droid;
17 using Firebase.Auth;
18
19 [assembly: Dependency(typeof(AuthDroid))]
20 namespace Application_Green_Quake.Droid
21 {
22     public class AuthDroid : IAuth
23     {
24         /** This function checks if a user is signed in.
25          */
26         public bool IsSignIn()
27         {
28             var user = FirebaseAuth.Instance.CurrentUser;
29             return user != null;
30
31         }
32         /** This function logs a user into the application
33          */
34         public async Task<string> LoginWithEmailAndPassword(string email, string password)
35         {
36             try
37             {
38                 var user = await
39                 FirebaseAuth.Instance.SignInWithEmailAndPasswordAsync(email, password);
39                 var token = user.Uid;
40                 return token;
41             }
42             catch (FirebaseAuthInvalidUserException e)
43             {
44                 e.PrintStackTrace();
45                 return string.Empty;
46             }
47             catch (FirebaseAuthInvalidCredentialsException e)
48             {
49                 e.PrintStackTrace();
50                 return string.Empty;
51             }
52         }
53         /** This function signs a user out of the application.
54          */
55         public bool SignOut()
56         {
57             try
58             {
59                 FirebaseAuth.Instance.SignOut();
60                 return true;
61             }
62         }
63     }
64 }
```

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

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

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

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

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

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