Technical Manual

Wildfire Location & Information App

30th April 2021

Name | Jack McNally

Student No | C00228758

Supervisor | Chris Meudec

Abstract

The purpose of the Wildfire project is to develop a mobile application for the General public and professional firefighters. This application should help people who fight fires as well as protect people who don’t. The application should be used as a tool to report fires, search for fires and notify people of nearby fires. It also should be able to provide the general public and the firefighters with useful information about fires and how to keep safe around them.

Table of Contents

Table of Figures

1. Introduction

The purpose of this technical manual is first to explain each of the individual screens from the wildfire application. I will describe the purpose of each screen, the navigation flow for each screen and what the components on each screen does. This will help display to the what is the purpose of each of the screens of the application.

The second section of this manual will finally display all of the project code. The code will be displayed as best as it can be shown on a document. The formatting of the code may be different as displayed on github or visual studio

If you would like to access the full project code please follow the link below to my GitHub account. Here you can view all of the commit throughout the year as well as the visual structure of my code.

Apk Link:
https://drive.google.com/file/d/1IJyQ5YBRvVnKv_1Y1z1QJzXu5GqSy92x/view?usp=sharing

GitHub Link:
https://github.com/mcnallyjack/Wildfire

Please be notified if you would like access to the API key or the firebase connection please send me an email on my personal or student email to request permission.

Personal Email: jackjjmcnally@gmail.com

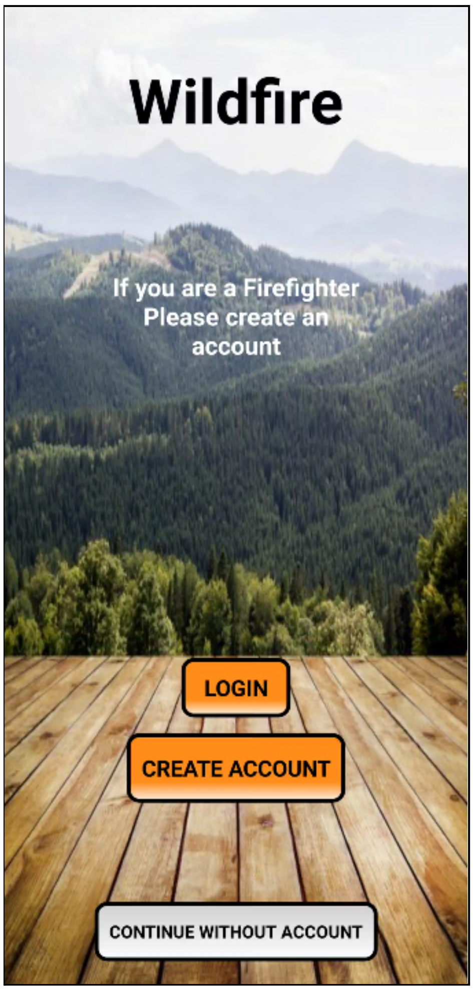
Student Email: c00228758@itcarlow.ie

The API keys have been replaced with the “API_KEY” keyword, the firebase connection has been replaced with “FB_CONN” & “FB_CONN”.

To download the application follow the link above, once the app has completed downloading, select open, you may then be prompted with a play protect screen, allow this. Now the application should open, please grant all permissions to use the application.

2. Application Screens

2.1. FirstPageView - Welcome Page



The purpose of this screen is to welcome the users to the Wildfire application. At the top of the screen the name of the application is stated along with a message aimed to professional firefighters.

The aim of this message is to ask the professional users to create an account.

Moving down the screen there are 3 options; Login, Create account and finally continue without account.

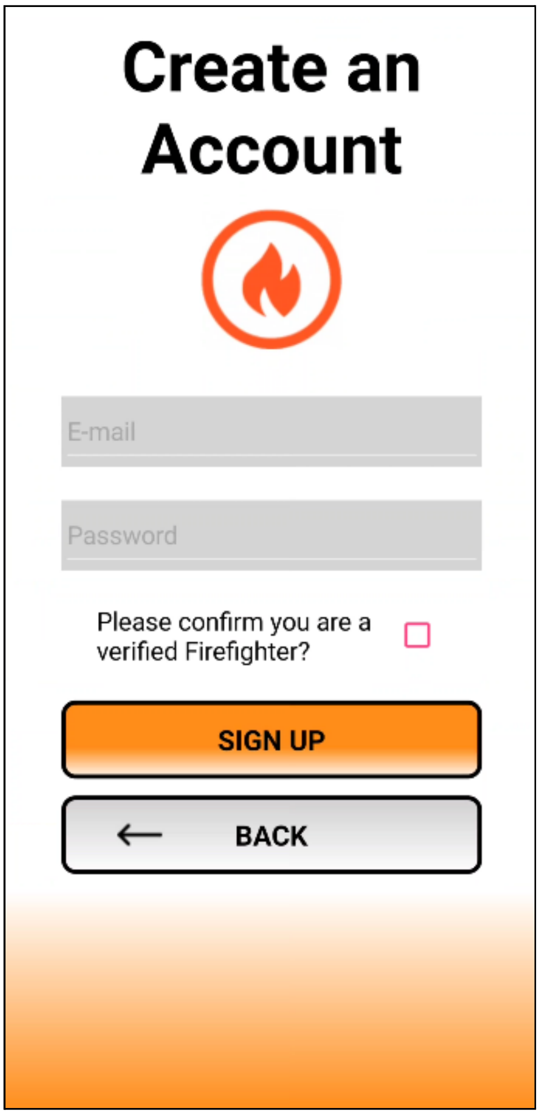
The **login** option is where continuing users will log into their account.

The **create account** option is where new professional firefighter users will create their account.

The **continue without account** option is where the general public users will gain access to the application.

Figure 1 - Welcome Screen.

2.2. SignUpView - Create Account Screen.



The purpose of the create account screen is to allow the professional to sign up for the application.

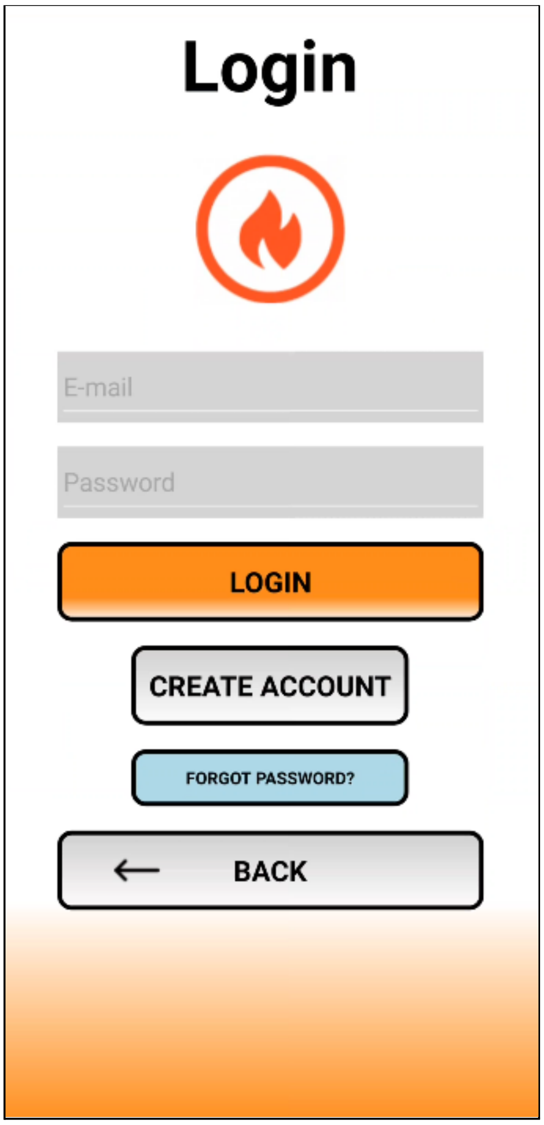
Beginning at the top of the screen the “Create an Account” message is displayed clearly as the heading of the screen. The logo of the application is then displayed followed by the entry fields for the users to input their email and password.

Once the user has entered their personal details the user must then select the checkbox that confirms if they are a verified firefighter or not.

Now that the user has entered all details correctly the may now select the **sign up** or **back** option. If the users sign up they will be directed to the login page and if they select back they will return to the previous page.

Figure 2 - Create an Account screen.

2.3. LoginPageView - Login Screen



The purpose of this screen is to allow the registered user to login the application.

From the top down the user first can see the page header “Login”. Moving down the users sees the application logo.

The user may now enter the credentials of their account. The user will first enter the email address associated with the account and then the password.

The user now has he following option;

Login - the user will select this option when all of the login credentials are correct. Once this is confirmed the user will be redirected to the view map screen.

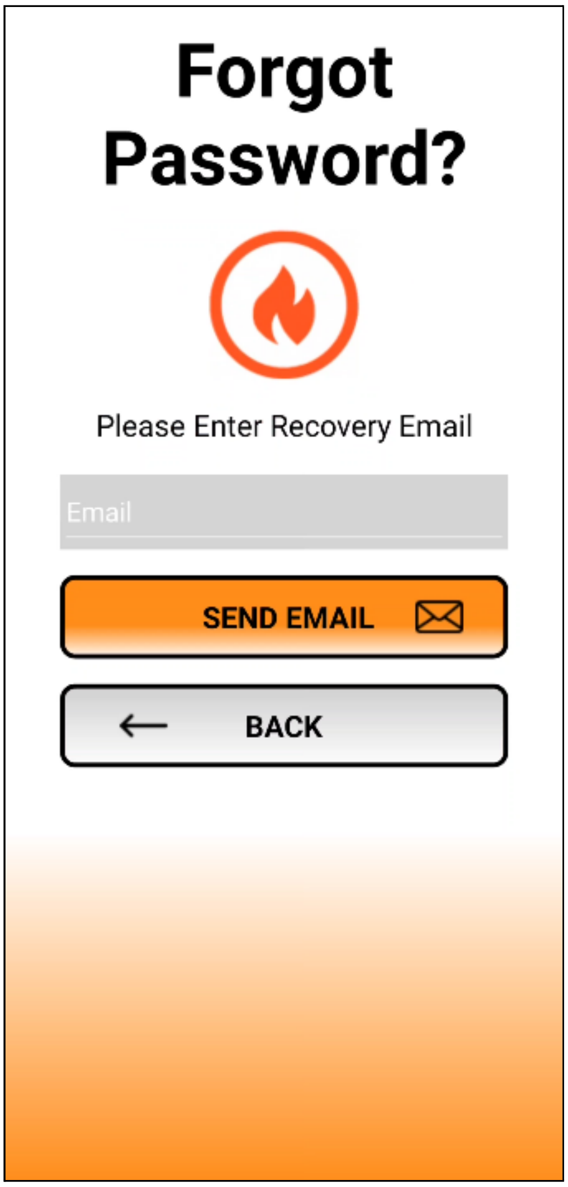
Create Account - the user will be directed to the sign up screen.

Forgot password - the user will be directed to the forgot password screen.

Back - the users will be directed to the previous screen.

Figure 3 - Login Screen.

2.4. ForgotPassword - Forgot password screen.



The purpose of this screen is to allow the user to reset their forgotten password.

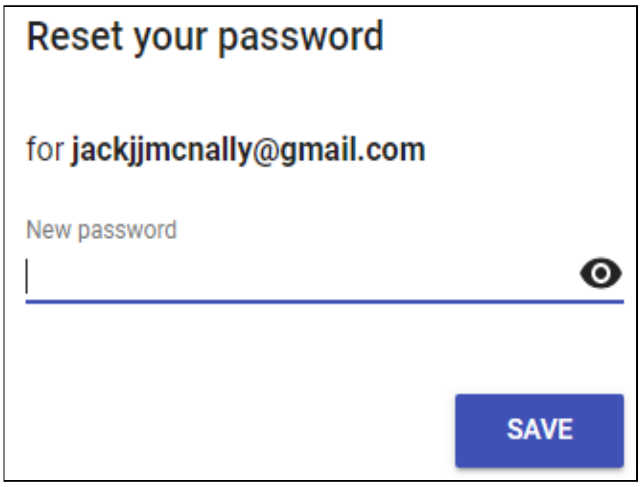
Beginning at the top of the screen the user is displayed the screen header “Forgot Password?”. Moving down the user is shown the application logo.

The user then must input the email to the account they have lost access to. Once the user has entered their email the user has two options.

Send email - this option will send a reset password link in an email for the user to enter their new password. Once the user selects this option they will be redirected to the login screen.

The second the option the user has is to select the back option this will redirect the user to the previous screen.

Figure 4 - Forgot Password Screen.



This is the screen that is displayed the the user once they have followed the link sent to their email address.

Figure 5 - Email Reset password.

2.5. MapView - View map screen

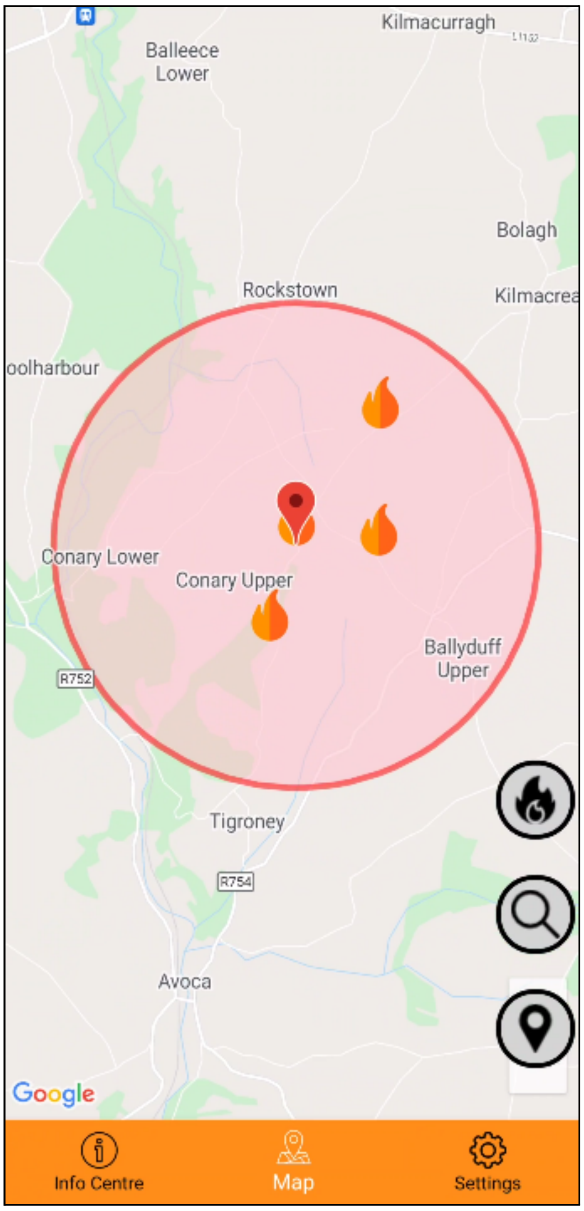


Figure 6 - View Map Screen.

The purpose of this screen is to display the map functionality to the user. All the users of the application once they pass the login section will come to this screen.

Beginning with the contents on the map, the first thing to point out is the red pin marker in the center of the red circle. This is the user's current location the user will have to grant the application permission to access this permission first.

The next element to point out is the red circle, this is the users radius circle that is set on the setting page. This circle acts as a barrier and if any fire penetrates that barrier the user will receive a notification if the user has enabled notification.

The final element to mention within the circle radius is the flame pins, these are the fires that have been reported by other users of the application.

Moving onto the options in the bottom right hand corner, these buttons have three different uses, the first reports a fire on the user's current location, the second opens the search functionality and the final button returns the user to their current location.

Finally the button navigation bar is available to the users and shows the three different tabs, the first is the info centre tab, the second tab which is the default option is the map view and the final tab is the settings.

The View map uses the long click event to report fires also, this is completed by long clicking on the location on the map itself will add a pin to the map and redirect you to the report fire screen.

2.6. ReportFireInfoView - Report fire screen.

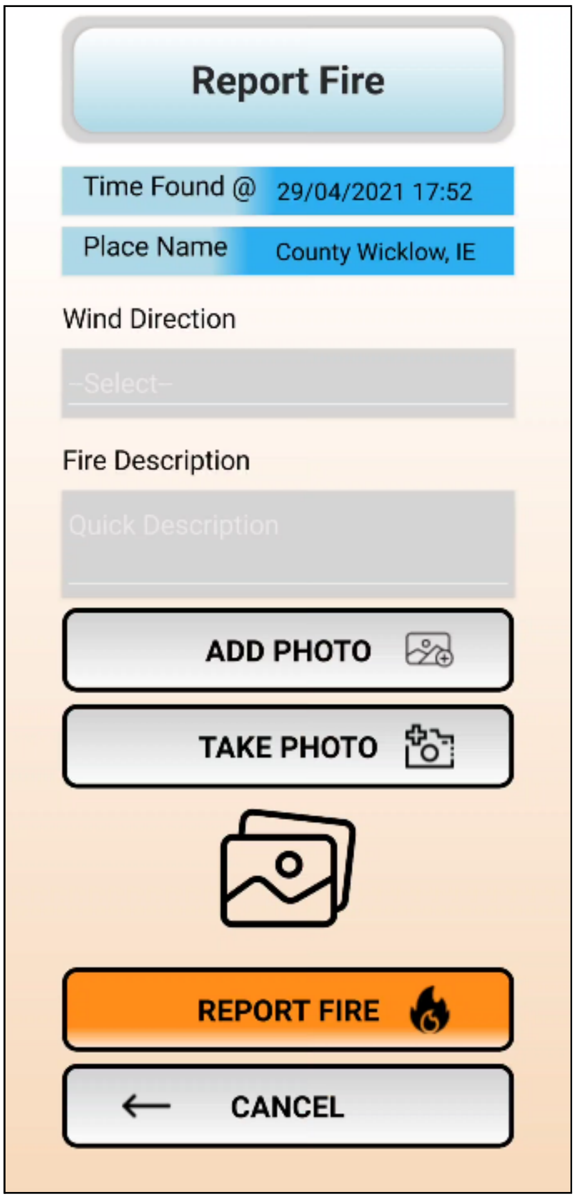


Figure 7 - Report fire screen.

The purpose of this screen is to display the report fire screen that the users of the application will use to report a fire to the map. This page is displayed either is a user long clicks on the map or use the button to report a fire on their current location.

Starting at the top of the page the user is greeted with the header “Report Fire”. The current time that the fore was reported as well as the place name. These two pieces of information are unchangeable.

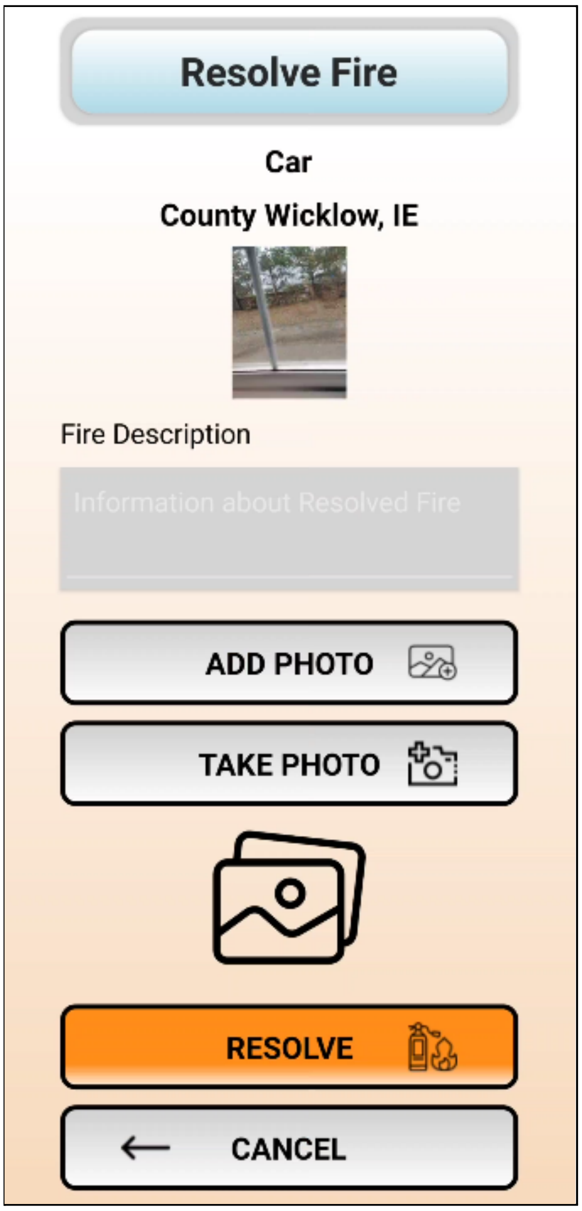
The user then has the option to enter the wind direction which is a drop down menu and the user will select their desired option.

Now the user will enter a description of the fire. This entry has a 100 character limit and must contain brief information.

The user then has the option to add a photo from their devices gallery or take a live photo. The photo then will replace the placeholder.

Once the has entered all the relevant information the user may now report the fire or select the cancel option. If the user reports the fire the fire will be added to the map if not the user will be returned to the view map screen.

2.7. ResolveFireInfoView - Resolve fire screen



The purpose of this screen is to resolve a current fire, this screen is only accessible by the professional user of this application. This page is accessed by selecting one of the fire pins on the view map screen.

Beginning at the top of the screen the user is displayed the page header “Resolve Fire”. Moving down the page there are two headings that display the fires description that was added during the report fire screen and also the location of the fire.

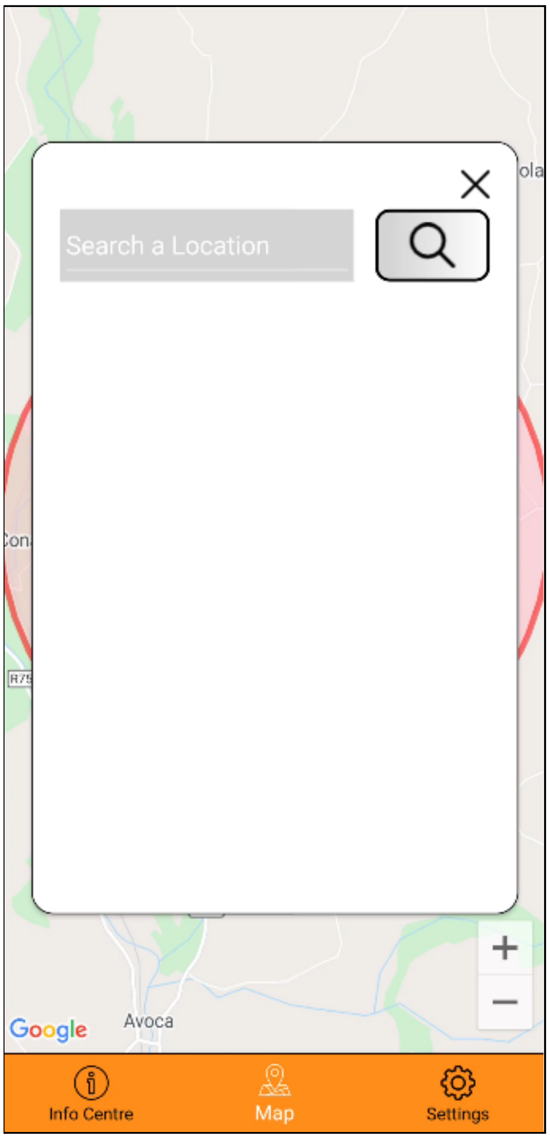
If an image was uploaded with the reported fire it is displayed next. The users now have to add an updated fire description that has a max character length of 100.

The users also have the option to add an updated photo either from their devices gallery or live camera. This photo will then be shown in the image placeholder.

Once a professional user has added all of the relevant information they can select the resolve fire option to resolve the fire, they will be redirected to the view map screen and the fire pin will be removed or they may select cancel to stop the fire being resolved.

Figure 8 - Resolve fire screen.

2.8. Search View Screen.



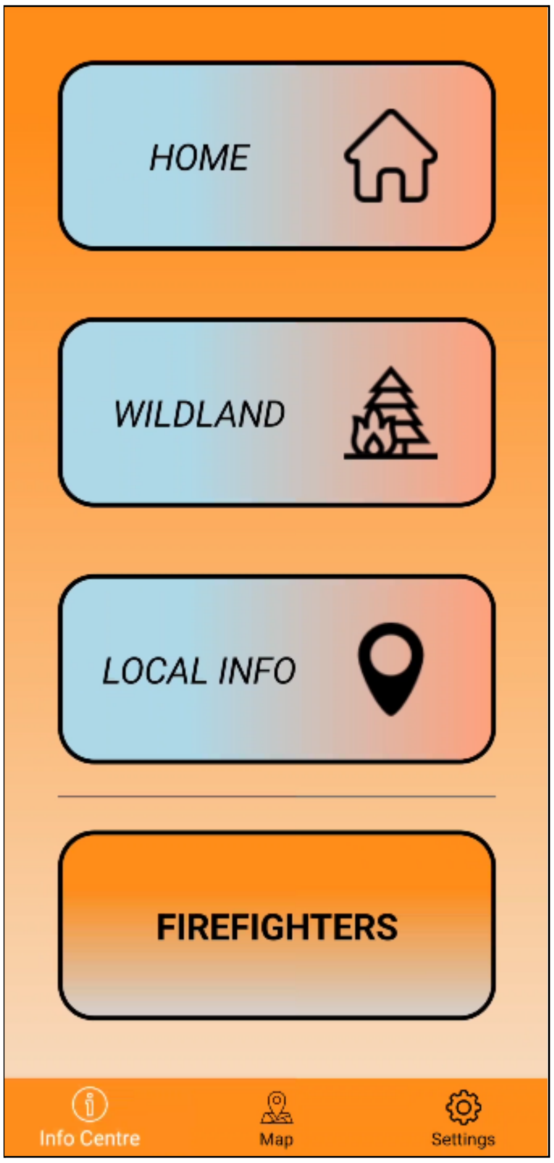
The purpose of this screen is to allow the users of the application search of a specific location on the map. This screen is housed within the map view screen and is accessible by selecting the search button on the view map screen.

To begin a search first the user must begin typing into the search location textbox. Once the user is finished typing the user will then select the magnifying glass icon and the screen will then move to the location on the map, this functionality will only work if the location is valid.

The user also has the option to return to the view map screen by selecting the cross in the top right corner of the search page.

Figure 9 - Search screen.

2.9. InfoView - Information centre screen.



The purpose of this screen is to allow the users of the application to view the included information.

Beginning at the top of the page, the first option the user can select is the home section. This section contains information to keep users safe from fires at home.

The next optionis the wildland section, this section aims to provide all users useful information if they are in a wildland fire situation.

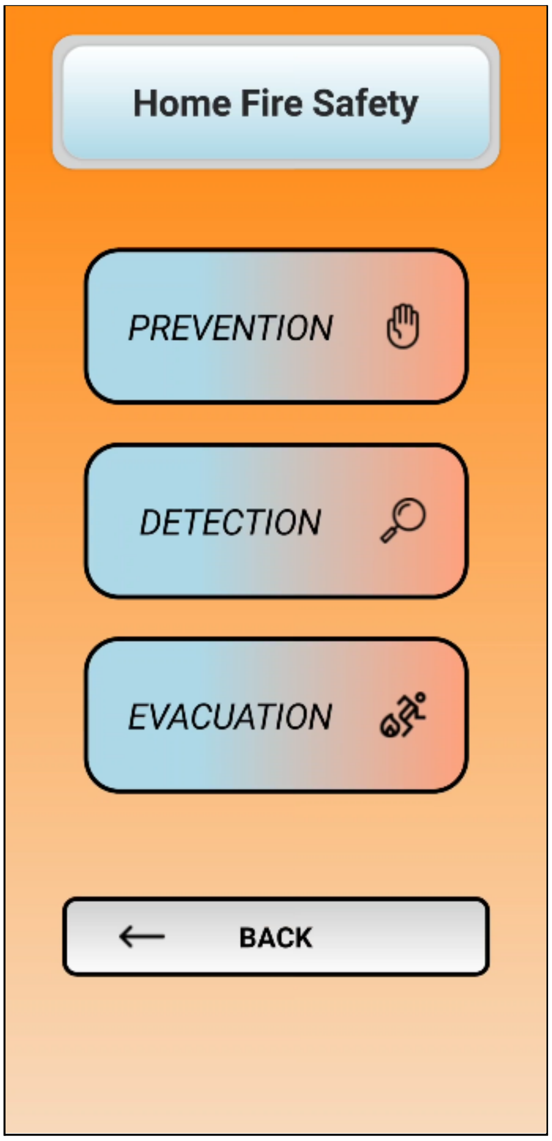
The next section which is accessible by al uses aims to help the users of the application gain some local information.

The final option on this page is the firefighters section; this section is an exclusive section which is only accessible to the professional users of the application. This section provides a list of all of the current and resolve fires.

The bottom section of this screen contains the navigation bar and can be used to quickly navigate to the other sections of the application.

Figure 10 - Information centre screen.

2.10. HomeInfoview - Home information centre



The purpose of this screen is to provide the user with the information regarding home fire safety.

This screen begins with a header that displays “” Home Fire Safety”.

This screen then has three different options that the user can select to view different information.

Prevention is the first section and helps the users understand how to prevent fires.

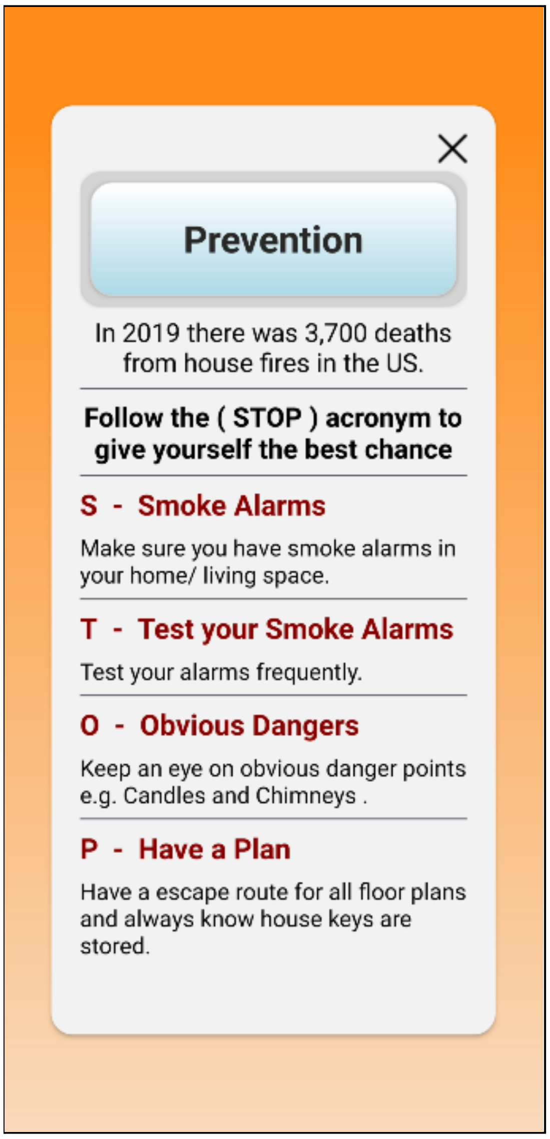
The next section is detection. This section contains information to help users detect fires.

The final option is evacuation and helps the user create a plan for evacuation.

At the bottom of the screen the user has the option to go back, this will redirect the user to the previous page.

Figure 11 - Home Info Centre.

2.11. HomeInfoView - Prevention Screen



Prevention

In 2019 there was 3,700 deaths from house fires in the US.

Follow the (STOP) acronym to give yourself the best chance

S - Smoke Alarms

Make sure you have smoke alarms in your home/ living space.

T - Test your Smoke Alarms

Test your alarms frequently.

O - Obvious Dangers

Keep an eye on obvious danger points e.g. Candles and Chimneys .

P - Have a Plan

Have a escape route for all floor plans and always know house keys are stored.

The purpose of this screen is to help the user understand how to prevent fires in the home situation.

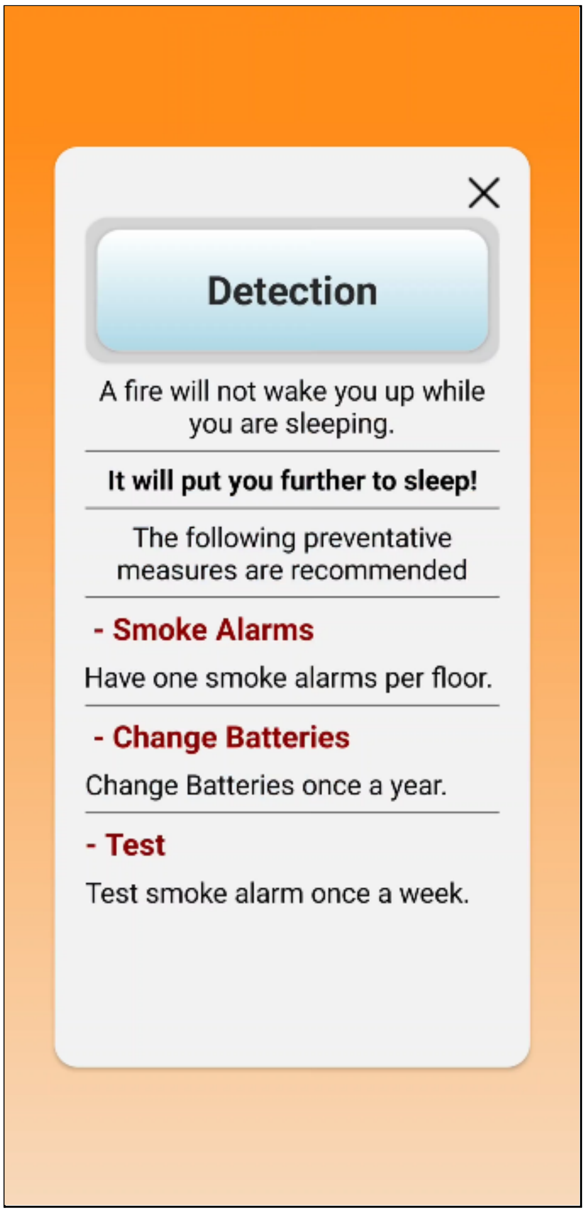
This screen first displays the users with the header “Prevention”.

The user can now read the information provided.

When the user is finished reading the screen the user can select the cross icon in the top right corner and will be redirected to the home info centre screen.

Figure 12 - HomeInfo prevention screen.

2.12. HomeInfoView - Detection



Detection

A fire will not wake you up while you are sleeping.

It will put you further to sleep!

The following preventative measures are recommended

- Smoke Alarms

Have one smoke alarms per floor.

- Change Batteries

Change Batteries once a year.

- Test

Test smoke alarm once a week.

The purpose of this screen is to provide the users of the application with the tips to improve the detection of fires within the home environment.

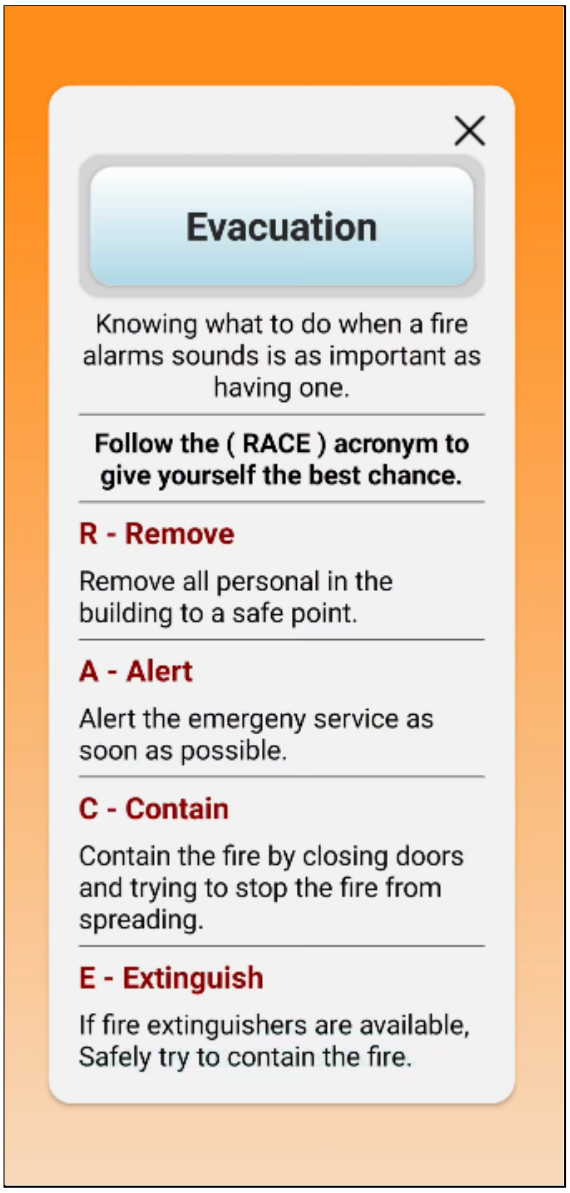
This screen first displays the users with the header "Detection".

The user can now read the information provided.

When the user is finished reading the screen the user can select the cross icon in the top right corner and will be redirected to the home info centre screen.

Figure 13 - HomeInfo Detection screen.

2.13. HomeInfoView - Evacuation.



Evacuation

Knowing what to do when a fire alarms sounds is as important as having one.

Follow the (RACE) acronym to give yourself the best chance.

R - Remove

Remove all personal in the building to a safe point.

A - Alert

Alert the emergeny service as soon as possible.

C - Contain

Contain the fire by closing doors and trying to stop the fire from spreading.

E - Extinguish

If fire extinguishers are available, Safely try to contain the fire.

The purpose of this screen is to provide the users of the application with the tips to safely evacuate a fire with the home environment.

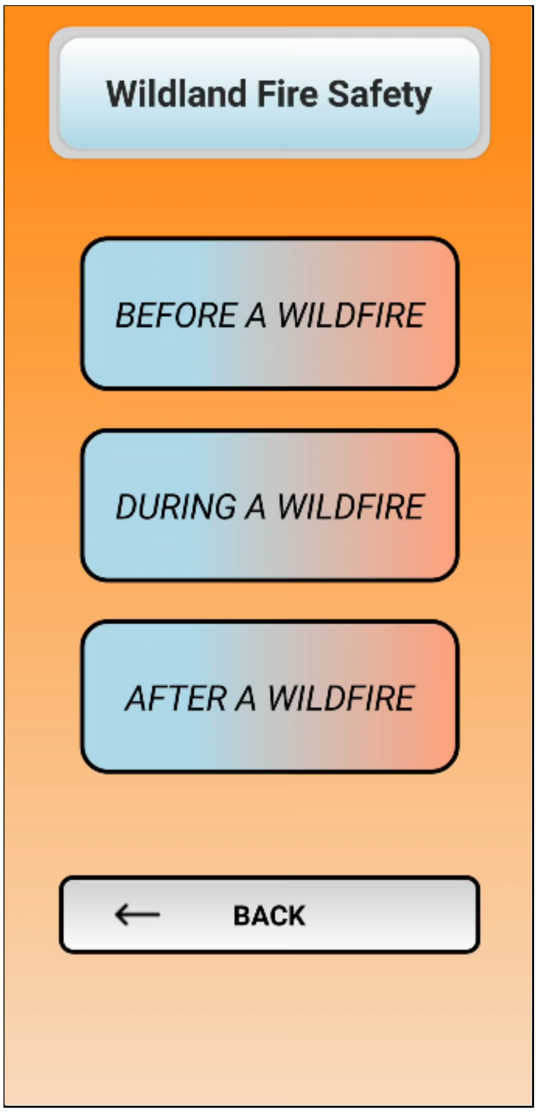
This screen first displays the users with the header “Evacuation”.

The user can now read the information provided.

When the user is finished reading the screen the user can select the cross icon in the top right corner and will be redirected to the home info centre screen.

Figure 14 - HomeInfo Evacuation Screen.

2.14. WildlandInfoView - Wildland information centre



The purpose of this screen is to aid the users of this application in the wildland environment.

This screen begins with a header that displays “” WildlandFire Safety”.

This screen then has three different options that the user can select to view different information.

Before a Wildfire; contains information about how to prepare for the threat of wildfires.

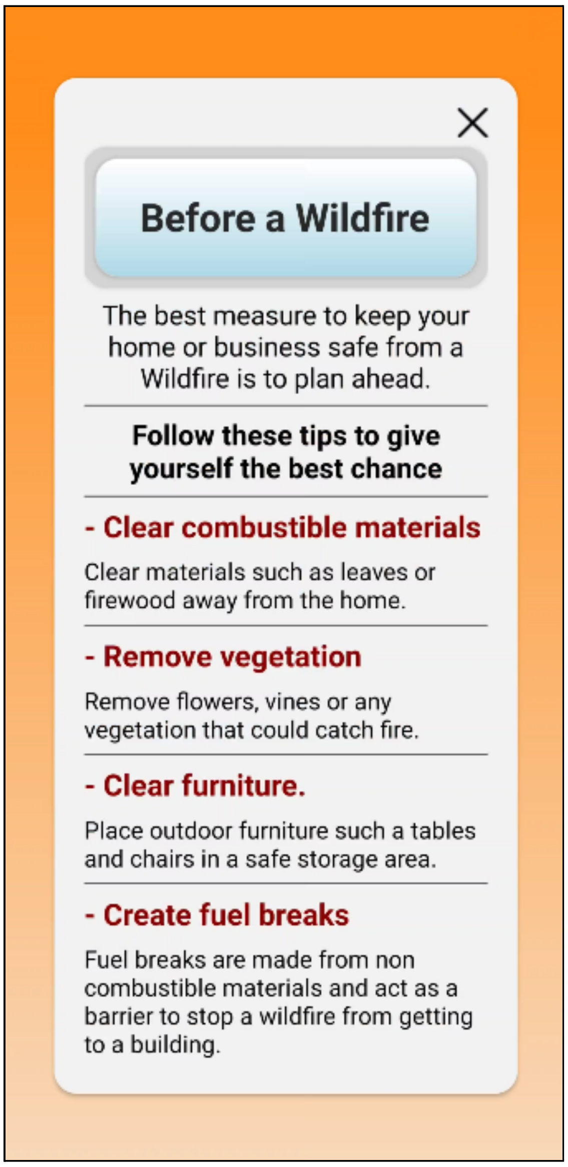
During a Wildfire; contains information how to keep safe during a wildfire.

After a Wildfire; contains information how to keep safe after a wildfire.

At the bottom of the screen the user has the option to go back, this will redirect the user to the previous page.

Figure 15 - Wildland Info centre.

2.15. WildlandInfoView - Before a Wildfire



Before a Wildfire

The best measure to keep your home or business safe from a Wildfire is to plan ahead.

Follow these tips to give yourself the best chance

- Clear combustible materials

Clear materials such as leaves or firewood away from the home.

- Remove vegetation

Remove flowers, vines or any vegetation that could catch fire.

- Clear furniture.

Place outdoor furniture such a tables and chairs in a safe storage area.

- Create fuel breaks

Fuel breaks are made from non combustible materials and act as a barrier to stop a wildfire from getting to a building.

The purpose of this screen is to provide the users of the application with useful tips in the event of wildfire threat.

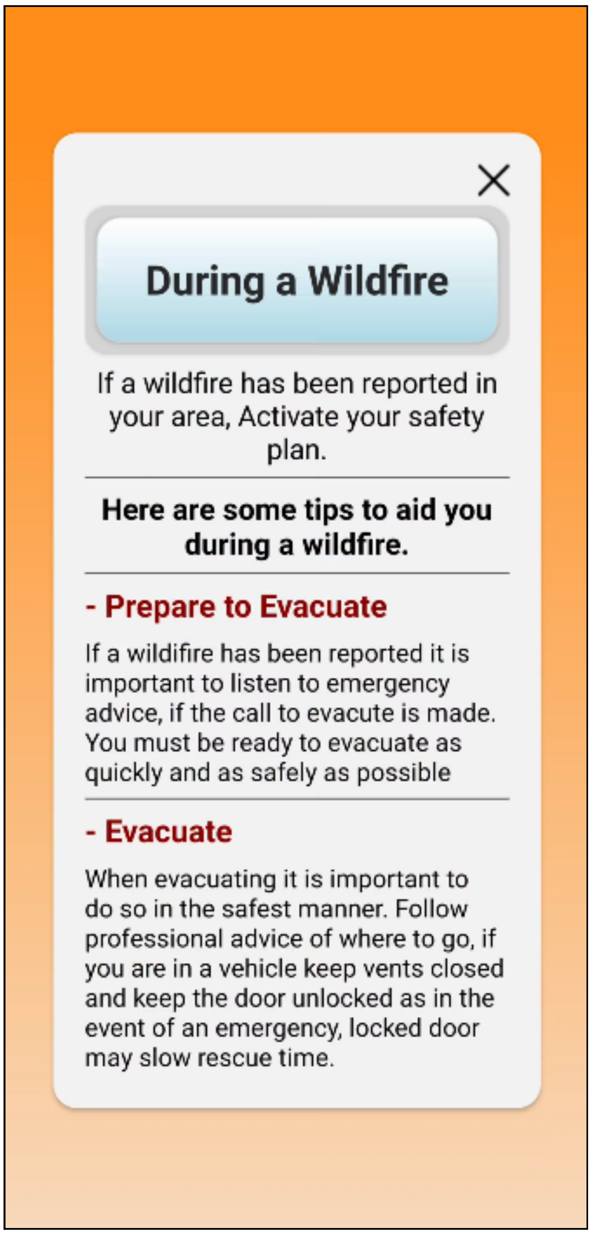
This screen first displays the users with the header “Before a Wildfire”.

The user can now read the information provided.

When the user is finished reading the screen the user can select the cross icon in the top right corner and will be redirected to the home info centre screen.

Figure 16 - WildlandInfo Before a Wildfire.

2.16. WildlandInfoview - During a Wildfire



During a Wildfire

If a wildfire has been reported in your area, Activate your safety plan.

Here are some tips to aid you during a wildfire.

- Prepare to Evacuate

If a wildfire has been reported it is important to listen to emergency advice, if the call to evacute is made. You must be ready to evacuate as quickly and as safely as possible

- Evacuate

When evacuating it is important to do so in the safest manner. Follow professional advice of where to go, if you are in a vehicle keep vents closed and keep the door unlocked as in the event of an emergency, locked door may slow rescue time.

The purpose of this screen is to aid the user of this application with useful tips in the event of a wildland fire.

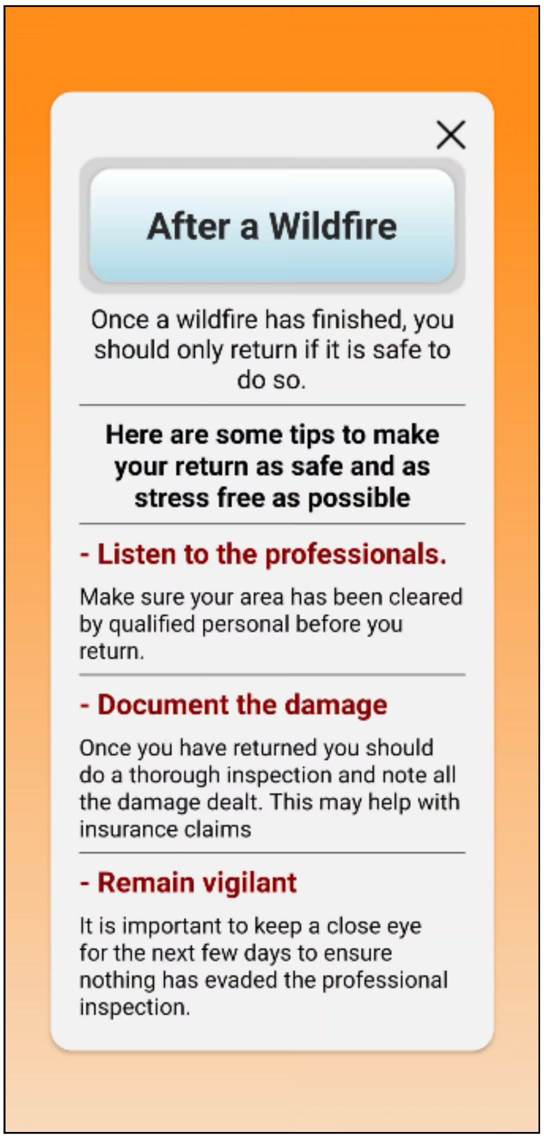
This screen first displays the users with the header “During a Wildfire”.

The user can now read the information provided.

When the user is finished reading the screen the user can select the cross icon in the top right corner and will be redirected to the home info centre screen.

Figure 17- WildlandInfo After a wildfire.

2.17. WildlandInfoView - After a Wildfire.



After a Wildfire

Once a wildfire has finished, you should only return if it is safe to do so.

Here are some tips to make your return as safe and as stress free as possible

- Listen to the professionals.

Make sure your area has been cleared by qualified personal before you return.

- Document the damage

Once you have returned you should do a thorough inspection and note all the damage dealt. This may help with insurance claims

- Remain vigilant

It is important to keep a close eye for the next few days to ensure nothing has evaded the professional inspection.

The purpose of this screen is to provide users of the application with some useful tips on what to do after a wildfire has occurred in their area.

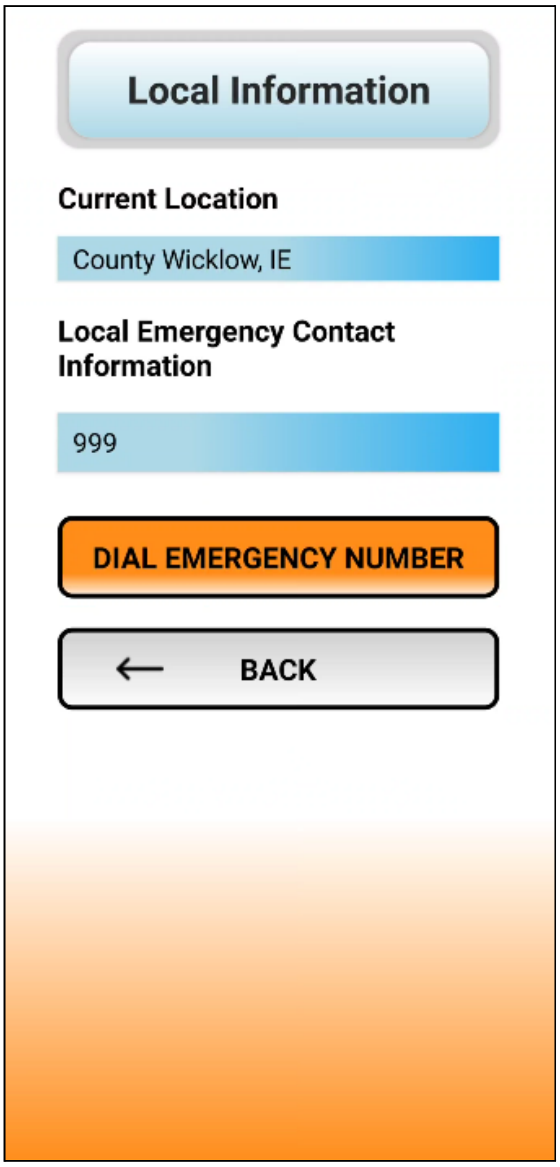
This screen first displays the users with the header “After a Wildfire”.

The user can now read the information provided.

When the user is finished reading the screen the user can select the cross icon in the top right corner and will be redirected to the home info centre screen.

Figure 18 - WildlandInfo After a wildfire.

2.18. LocalInfoView - Local information.



The purpose of this screen is to provide the users of the application with some local information about their current location.

Beginning at the top of the screen the user is first displayed the “Local Information header”.

Moving down the screen the name of the current location is displayed to the user.

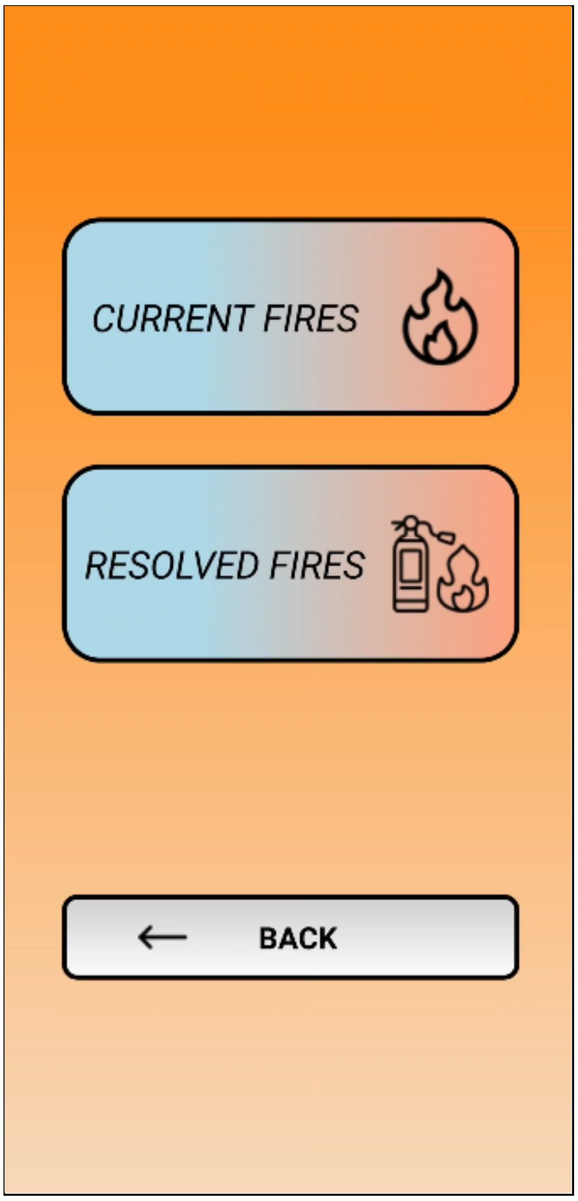
Next the emergency contact information of the user current location is also displayed.

The final two options on the screen are buttons, first the user has the option to dial the emergency number provided and once the user selects this option the user will be redirected to the device dialer.

Finally the last option is the back button and will redirect the user to the previous screen.

Figure 19 - Local Information.

2.19. ProFireInfoView - Firefighter screen



The main purpose of this screen is to exclusively provide the professional users of this application with information regarding the current and resolved fires.

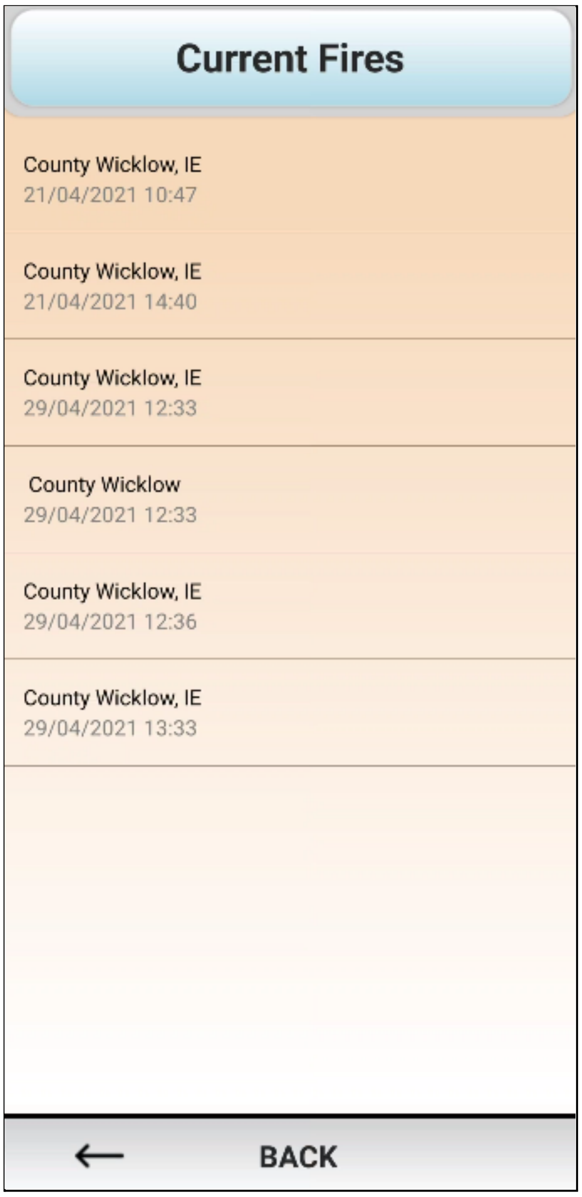
There are two main options on this screen: the first option is a button that redirects the user to a list of the current fires on the application. On this screen the professional user can select and fire and view the fire details.

The second option is a button that redirects the professional users to the application to a list of the resolved fires on the application. On this screen the professional user can select a fire and view its details.

The final option on this page is the back button. This button will redirect the user to the previous page.

Figure 20 - ProInfoView firefighter screen.

2.20. ProInfoFireView - Current Fires



Current Fires

County Wicklow, IE
21/04/2021 10:47

County Wicklow, IE
21/04/2021 14:40

County Wicklow, IE
29/04/2021 12:33

County Wicklow
29/04/2021 12:33

County Wicklow, IE
29/04/2021 12:36

County Wicklow, IE
29/04/2021 13:33

The purpose of this screen is to provide the professional users of this application a list of the current fires.

Beginning at the top of the screen the user iis displayed the header “Current Fires”.

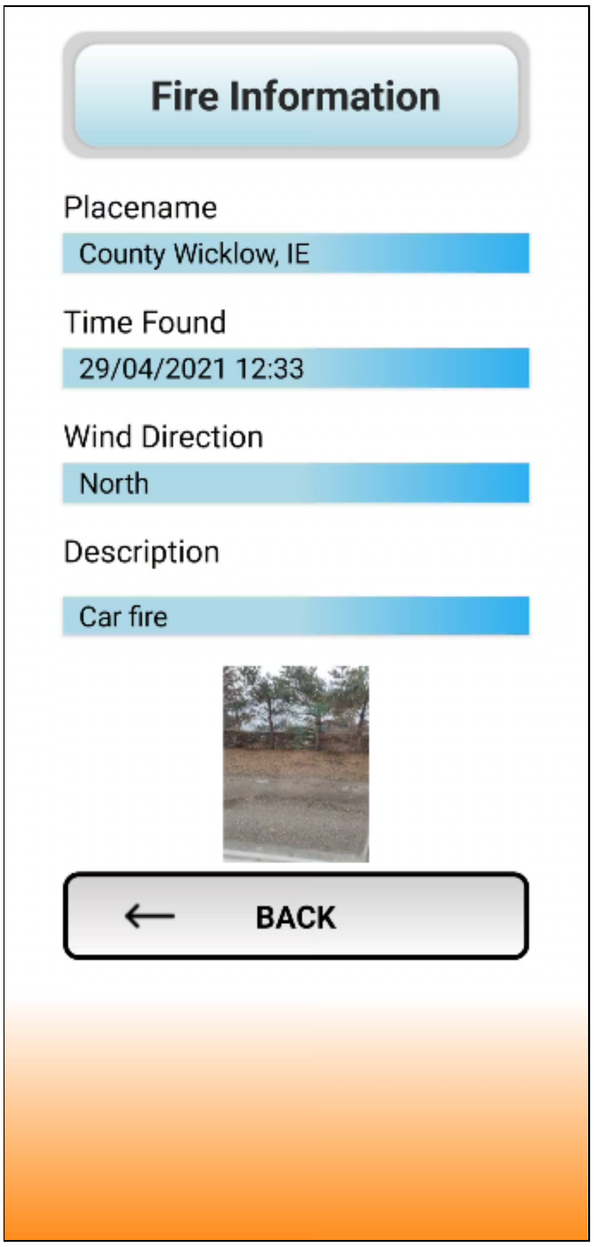
Moving down the screen the user is displayed a list of the current fires that are currently reported on the application.

The professional users can select the fire that they wish to view by simply selecting the fire. Once the user selects the fire the user will be redirected to the selected view screen.

The final option for the user on this screen is to select the back option, this will redirect the user to the previous screen.

Figure 21 - ProInfoView current fires.

2.21. ProInfoFireView - Current selected fire



The purpose of this screen is to provide the professional users of the application with the information on current fires on the application.

Beginning at the top of the page the professional user is displayed the header “Fire Information”.

Moving down the screen the user is displayed some details about the fire;

Placename, this shows the current location of the fire.

Time Found, this displays the time the fire was reported.

Wind direction, this displays the wind direction when the fire was reported.

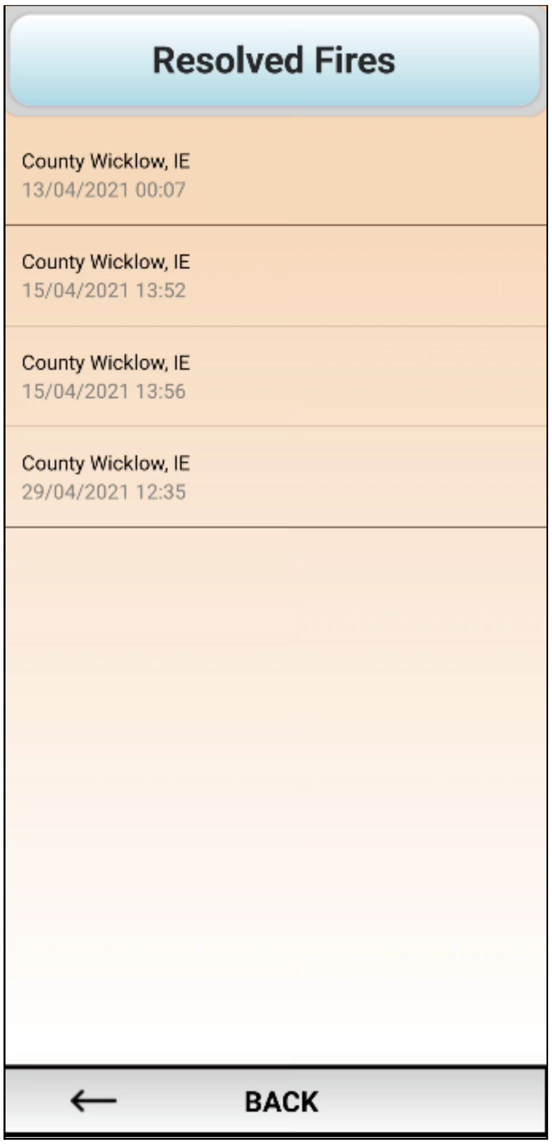
Description, this is the description that was provided by the user when the fire was reported.

Finally an image is provided that the user uploaded.

The last section of the screen contains the back button that redirects the user to the previous screen.

Figure 22 - ProInfoView selected current fire.

2.22. ProInfoFireView - Resolved fires



County Wicklow, IE
13/04/2021 00:07

County Wicklow, IE
15/04/2021 13:52

County Wicklow, IE
15/04/2021 13:56

County Wicklow, IE
29/04/2021 12:35

The purpose of this screen is to provide the professional users of this application a list of the resolved fires.

Beginning at the top of the screen the user iis displayed the header “Resolved Fires”.

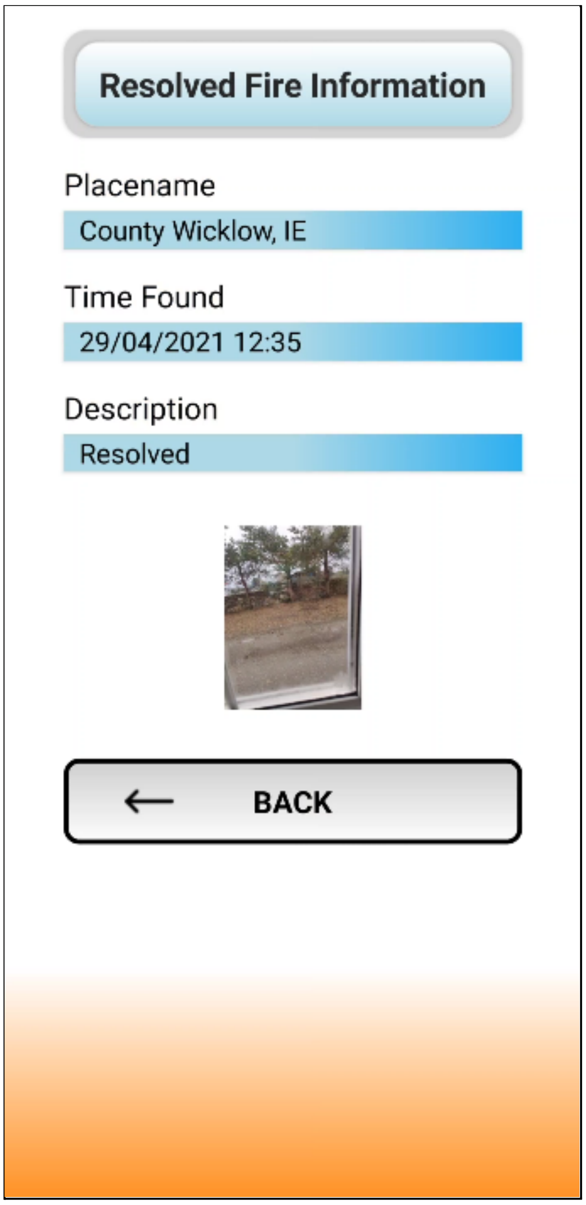
Moving down the screen the user is displayed a list of the resolved fires that are currently resolvedon the application.

The professional users can select the fire that they wish to view by simply selecting the fire. Once the user selects the fire the user will be redirected to the selected view screen.

The final option for the user on this screen is to select the back option, this will redirect the user to the previous screen.

Figure 23 - ProInfoView Resolved fires.

2.23. ProInfoFireView - Resolved selected fire.


Resolved Fire Information

Placename

County Wicklow, IE

Time Found

29/04/2021 12:35

Description

Resolved

The purpose of this screen is to provide the professional users of the application with the information on resolved fires on the application.

Beginning at the top of the page the professional user is displayed the header “Resolved Fire Information”.

Moving down the screen the user is displayed some details about the fire;

Placename, this shows the current location of the fire.

Time Found, this displays the time the fire was reported.

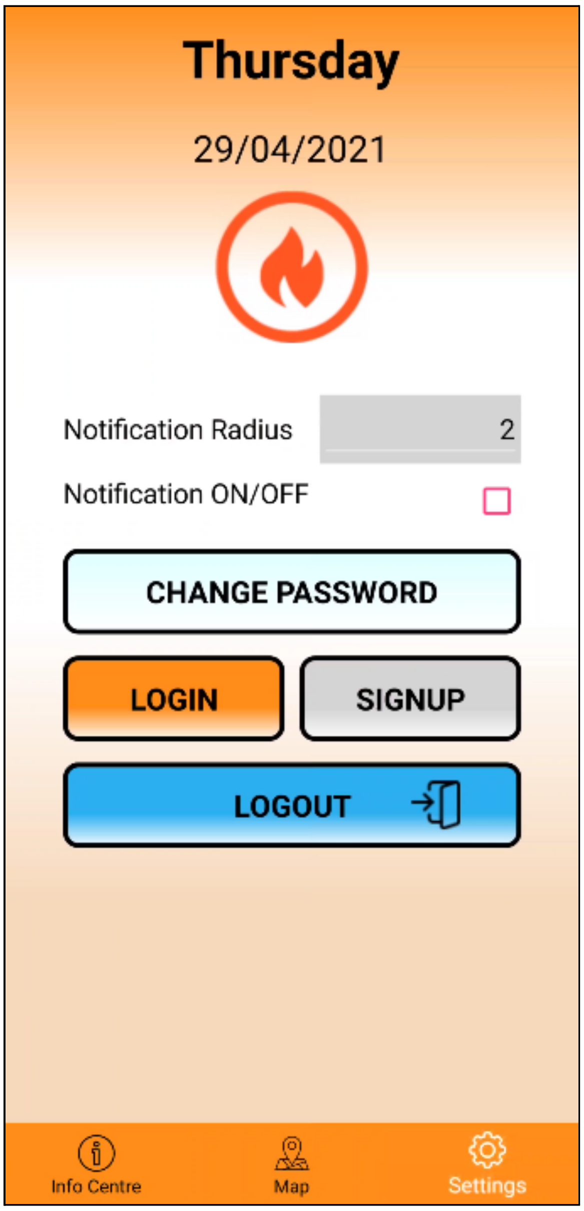
Description, this is the description that was provided by the user when the fire was reported.

Finally an image is provided that the user uploaded.

The last section of the screen contains the back button that redirects the user to the previous screen.

Figure 24 - ProInfoView selected resolved fires.

2.24. Settings View



The purpose of this screen is to provide all users of the application with a settings page.

Beginning at the top of the page the user is provided the current day and date.

The logo of the application is then displayed to the user.

The next two sections are key functionalities. The notification radius is the first setting here the user will be able to set a radius around their current location to their desired distance. In this case the radius is 2km. This can be seen on the view map screen. The next key section on this screen is the notification preferences, here the user of the application can turn on and off the notifications.

These notifications are powered by the notification radius if it is set and if it is not set there is a default radius of 5km.

The next section of this screen contains buttons. The first button will redirect the user to the change password screen inorder to change their current password.

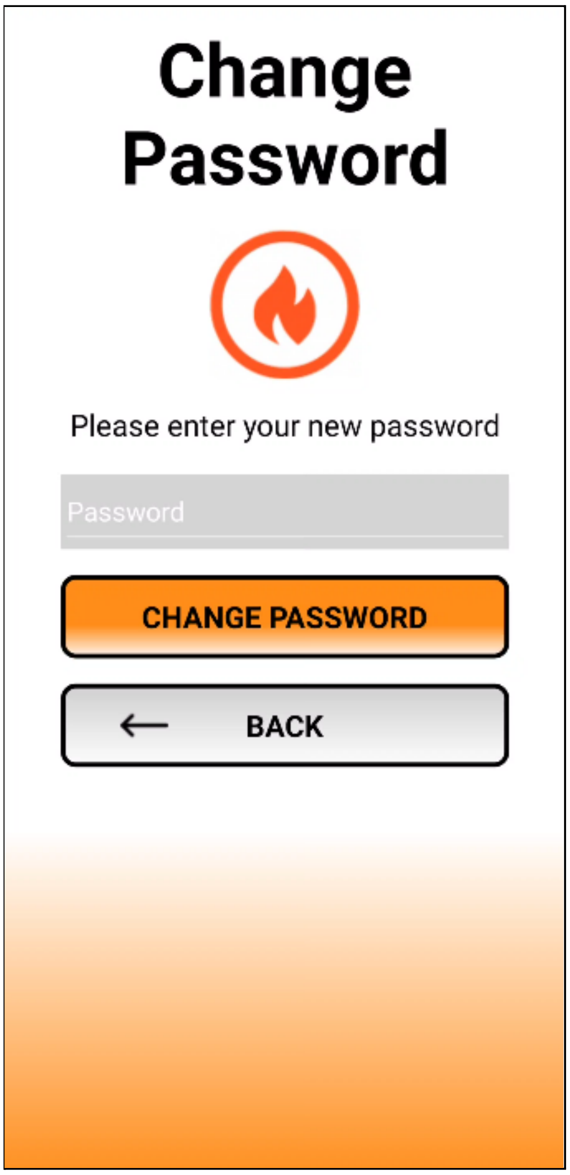
Next the login and sign up buttons are displayed, these will redirect the user to the login screen and sign up screen respectfully.

Figure 25 - Setting Screen.

The last option on this screen allows the user to logout, this will end the users session and redirect the user to the first view screen.

The last element on this screen contains the navigation bar that allows the user of the application to quickly navigate to the info centre screen of the view map screen.

2.25. Change Password



The purpose of this screen is to allow the professional users of the application to change their password.

Beginning at the top of the page the user is displayed the “Change Password” header.

Moving down the application logo is displayed. The password entry is then displayed here the user will be asked to enter the new password they would use to keep their account secure.

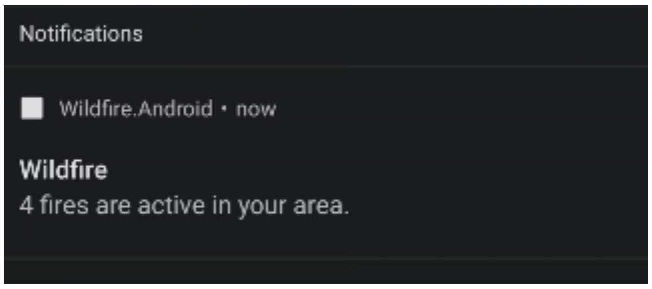
Once the professional user has entered their password they will then select either of the final options.

If the professional user selects the change password option the professional users password will be changed is it meets certain criteria. The user will then be redirected to the settings page.

If the users selects the back option the user will be redirected to their current page.

Figure 26 - Change password screen.

2.26. Notifications


Notifications

Wildfire.Android • now

Wildfire
4 fires are active in your area.

The purpose of this screen is to display an example of the notification that the user of the application may receive.

The user may select this notification and will be redirected to the view map screen.

Figure 27 - Notification screen.

3. Project Code

3.1. Wildfire

3.1.1. Helper

3.1.1.1. Firebase Helper.cs

```
/* Author:      Jack McNally
* Page Name:   FirebaseHelper
* Purpose:     All Firebase helper logic.
*/
using System;
using System.Collections.Generic;
using System.Text;
using Wildfire.Models;
using Firebase.Database;
using Firebase.Database.Query;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;


namespace Wildfire.Helper
{
   public class FirebaseHelper
   {
       FirebaseClient firebase = new
FirebaseClient("https://driven-bulwark-297919-default-rtdb.firebaseio.com/");

       // Get all fires
       public async Task<List<Fire>> GetAllFires()
       {
           return (await firebase
               .Child("Fire")
               .OnceAsync<Fire>()).Select(item => new Fire
               {
                   FireID = item.Object.FireID,
                   Latitude = item.Object.Latitude,
                   Longitude = item.Object.Longitude,
                   Time = item.Object.Time,
                   WindDirection = item.Object.WindDirection,
                   Description = item.Object.Description,
                   PlaceName = item.Object.PlaceName,
                   ResolvedDescription = item.Object.ResolvedDescription,
                   DeviceID = item.Object.DeviceID
               }).ToList();
       }

       // Get all resolved fires
       public async Task<List<Fire>> GetResolvedFires()
```

```
        {
            return (await firebase
                .Child("ResolvedFire")
                .OnceAsync<Fire>()).Select(item => new Fire
                {
                    FireID = item.Object.FireID,
                    PlaceName = item.Object.PlaceName,
                    Time = item.Object.Time,
                    ResolvedDescription = item.Object.ResolvedDescription,

                }).ToList();
        }

        // Add a fire
        public async Task AddFire(string fireId, string lat, string longi, string time,
string direction, string desc, string plaName, string deviceID)
        {
            await firebase
                .Child("Fire")
                .PostAsync(new Fire() { FireID = fireId, Latitude = lat, Longitude =
longi, Time=time, WindDirection = direction, Description = desc, PlaceName = plaName,
DeviceID = deviceID});
        }

        // Resolve a fire
        public async Task ResolveFire(string fireId)
        {
            var toResolve = (await firebase
                .Child("Fire")
                .OnceAsync<Fire>()).Where(a => a.Object.FireID ==
fireId).FirstOrDefault();
            await firebase.Child("Fire").Child(toResolve.Key).DeleteAsync();
        }

        // Add Resolves a fire
        public async Task AddResolvedFire(string fireId, string desc, string place,
string newDesc, string time)
        {
            await firebase
                .Child("ResolvedFire")
                .PostAsync(new Fire() { FireID = fireId, Description = desc, PlaceName =
place, ResolvedDescription = newDesc, Time = time });
        }

        // Get a fire
        public async Task<Fire> GetFire(string fireId)
        {
            var allFires = await GetAllFires();
            await firebase
                .Child("Fire")
                .OnceAsync<Fire>();
            return allFires.Where(a => a.FireID == fireId).FirstOrDefault();
        }
    }
```

```
}
```

Figure 28 - FirebaseHelper.cs

3.1.2. Models

3.1.2.1. Fire.cs

```
/* Author:      Jack McNally
* Page Name:   Fire
* Purpose:     All information for fire reporting.
*/
using System;
using System.Collections.Generic;
using System.Text;

namespace Wildfire.Models
{
   public class Fire
   {
       public string FireID { get; set; }
       public string Latitude { get; set; }
       public string Longitude { get; set; }
       public string Time { get; set; }
       public string WindDirection { get; set; }
       public string Description { get; set; }
       public string PlaceName { get; set; }
       public string ResolvedDescription { get; set; }
       public string DeviceID { get; set; }

   }
}
```

Figure 29 - Fire.cs

3.1.2.2. Google Place.cs

```
/* Author:      Jack McNally
* Page Name:   GooglePlace
* Purpose:     used for autocomplete search functionality.
*/
using System;
using System.Collections.Generic;
using System.Text;
using Newtonsoft.Json.Linq;

namespace Wildfire.Models
{
   public class GooglePlace
   {
       public string Name { get; set; }
       public double Latitude { get; set; }
       public double Longitude { get; set; }
```

```
        public string Raw { get; set; }

        public GooglePlace(JObject jsonObject)
        {
            Name = (string)jsonObject["result"]["name"];
            Latitude = (double)jsonObject["result"]["geometry"]["location"]["lat"];
            Longitude = (double)jsonObject["result"]["geometry"]["location"]["lng"];
            Raw = jsonObject.ToString();
        }
    }
}
```

Figure 30 - GooglePlace.cs

3.1.2.3. GooglePlaceAutoCompletePrediction.cs

```
/* Author:      Jack McNally
* Page Name:   GooglePlaceAutoCompletePrediction
* Purpose:     GooglePlaceAutoCompletePrediction functionality.
*/
using System;
using System.Collections.Generic;
using Newtonsoft.Json;

namespace Wildfire.Models
{
    public class GooglePlaceAutoCompletePrediction
    {
        [JsonProperty("description")]
        public string Description { get; set; }

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

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

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

        [JsonProperty("structured_formatting")]
        public StructuredFormatting StructuredFormatting { get; set; }

    }

    public class StructuredFormatting
    {
        [JsonProperty("main_text")]
        public string MainText { get; set; }

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

    public class GooglePlaceAutoCompleteResult
    {
        [JsonProperty("status")]
        public string Status { get; set; }

        [JsonProperty("predictions")]
        public List<GooglePlaceAutoCompletePrediction> AutoCompletePlaces { get; set; }
    }
}
```

Figure 31 - GooglePlaceAutoCompletePrediction.cs

3.1.3. Services

3.1.3.1. GoogleMapsApiService.cs

```
/* Author:      Jack McNally
* Page Name:   GoogleMapsApiService
* Purpose:     Google map api implementation
*/
using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using Wildfire.Models;

namespace Wildfire.Services
{
    public class GoogleMapsApiService : IGoogleMapsApiService
    {
        static string _googleMapsKey;

        private const string ApiBaseAddress = "https://maps.googleapis.com/maps/";
        private HttpClient CreateClient()
        {
            var httpClient = new HttpClient
            {
                BaseAddress = new Uri(ApiBaseAddress)
            };

            httpClient.DefaultRequestHeaders.Accept.Clear();
            httpClient.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

            return httpClient;
        }
        public static void Initialize(string googleMapsKey)
        {
            _googleMapsKey = googleMapsKey;
        }

        // GetPlaces
        public async Task<GooglePlaceAutoCompleteResult> GetPlaces(string text)
        {
            GooglePlaceAutoCompleteResult results = null;

            using (var httpClient = CreateClient())
            {
                var response = await
httpClient.GetAsync($"api/place/autocomplete/json?input={Uri.EscapeUriString(text)}&key=
{_googleMapsKey}").ConfigureAwait(false);
                if (response.IsSuccessStatusCode)
                {
                    var json = await
response.Content.ReadAsStringAsync().ConfigureAwait(false);
                    if (!string.IsNullOrWhiteSpace(json) && json != "ERROR")
                    {
                        results = await Task.Run(() =>

JsonConvert.DeserializeObject<GooglePlaceAutoCompleteResult>(json)
                        ).ConfigureAwait(false);

                    }
                }
            }

            return results;
        }

        // GetPlaceDetails
        public async Task<GooglePlace> GetPlaceDetails(string placeId)
        {
            GooglePlace result = null;
            using (var httpClient = CreateClient())
            {
                var response = await
httpClient.GetAsync($"api/place/details/json?placeid={Uri.EscapeUriString(placeId)}&key=
{_googleMapsKey}").ConfigureAwait(false);
                if (response.IsSuccessStatusCode)
                {
                    var json = await
response.Content.ReadAsStringAsync().ConfigureAwait(false);
                    if (!string.IsNullOrWhiteSpace(json) && json != "ERROR")
                    {
                        result = new GooglePlace(JObject.Parse(json));
                    }
                }
            }

            return result;
        }
    }
}
```

Figure 32 - GoogleMapsApiService.cs

3.1.3.2. IAuth.cs

```
/* Author:      Jack McNally
 * Page Name:   IAuth
 * Purpose:     Used for authentication.
 */
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

namespace Wildfire.Services
{
   public interface IAuth
   {
       Task<string> LoginWithEmailAndPassword(string email, string password);

       Task<string> SignUpWithEmailAndPassword(string email, string password);

       Task ForgotPassword(string email);

       Task ChangePassword(string newPassword);

       bool SignOut();

       bool SignIn();


   }
}
```

Figure 33 - IAuth.cs

3.1.3.3. IGoogleMapsApiService.cs

```
/* Author:      Jack McNally
 * Page Name:   IGoogleMapsApiService
 * Purpose:     IGoogleMapsApiService for places.
 */
using System.Threading.Tasks;
using Wildfire.Models;

namespace Wildfire.Services
{
   public interface IGoogleMapsApiService
   {
       Task<GooglePlaceAutoCompleteResult> GetPlaces(string text);
       Task<GooglePlace> GetPlaceDetails(string placeId);
   }
}
```

Figure 34 - IGoogleMapsApiService.cs

3.1.3.4. INotification.cs

```
/* Author:      Jack McNally
* Page Name:   INotification
* Purpose:     notification service.
*/
using System;
using System.Collections.Generic;
using System.Text;

namespace Wildfire.Services
{
   public interface INotification
   {
       void CreateNotification(String title, String message);
   }
}
```

Figure 35 - INotification.cs

3.1.4. ViewModels

3.1.4.1. MainViewModel.cs

```
/* Author:      Jack McNally
* Page Name:   MainViewModel
* Purpose:     Functionality for search place.
*/
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Input;
using Wildfire.Views;
using Wildfire.Models;
using Wildfire.Services;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.GoogleMaps;

namespace Wildfire.ViewModels
{
   public class MainViewModel : INotifyPropertyChanged
   {
       public ICommand CalculateRouteCommand { get; set; }
       public ICommand UpdatePositionCommand { get; set; }

       public ICommand LoadRouteCommand { get; set; }
       public ICommand StopRouteCommand { get; set; }
       IGoogleMapsApiService googleMapsApi = new GoogleMapsApiService();
```

```
        public bool _hasRouteRunning { get; set; }
        string _originLatitud;
        string _originLongitud;
        string _destinationLatitud;
        string _destinationLongitud;

        GooglePlaceAutoCompletePrediction _placeSelected;
        public GooglePlaceAutoCompletePrediction PlaceSelected
        {
            get
            {
                return _placeSelected;
            }
            set
            {
                _placeSelected = value;
                if (_placeSelected != null)
                    GetPlaceDetailCommand.Execute(_placeSelected);
            }
        }
        public ICommand FocusOriginCommand { get; set; }
        public ICommand GetPlacesCommand { get; set; }
        public ICommand GetPlaceDetailCommand { get; set; }

        public ObservableCollection<GooglePlaceAutoCompletePrediction> Places { get; set;
}
        public ObservableCollection<GooglePlaceAutoCompletePrediction> RecentPlaces {
get; set; } = new ObservableCollection<GooglePlaceAutoCompletePrediction>();

        public bool ShowRecentPlaces { get; set; }
        bool _isPickupFocused = true;

        string _pickupText;
        public string PickupText
        {
            get
            {
                return _pickupText;
            }
            set
            {
                try
                {
                    _pickupText = value;
                    if (!string.IsNullOrEmpty(_pickupText))
                    {
                        _isPickupFocused = true;
                        GetPlacesCommand.Execute(_pickupText);
                    }
                    else
                    {

                    }
                }
                catch(Exception ex)
                {
                    ex.Message.ToString();
                }
            }
        }

        string _originText;
        public string OriginText
        {
            get
            {
                return _originText;
            }
            set
            {
                _originText = value;
                if (!string.IsNullOrEmpty(_originText))
                {
                    _isPickupFocused = false;
                    GetPlacesCommand.Execute(_originText);
                }
            }
        }
         public MainViewModel()
         {

            StopRouteCommand = new Command(StopRoute);
            GetPlacesCommand = new Command<string>(async (param) => await
GetPlacesByName(param));
            GetPlaceDetailCommand = new Command<GooglePlaceAutoCompletePrediction>(async
(param) => await GetPlacesDetail(param));

         }

        public void StopRoute()
        {
            _hasRouteRunning = false;
        }

        public async Task GetPlacesByName(string placeText)
        {
            var places = await googleMapsApi.GetPlaces(placeText);
            var placeResult = places.AutoCompletePlaces;
            if (placeResult != null && placeResult.Count > 0)
            {
                Places = new
ObservableCollection<GooglePlaceAutoCompletePrediction>(placeResult);
            }

            ShowRecentPlaces = (placeResult == null || placeResult.Count == 0);
        }

        public async Task GetPlacesDetail(GooglePlaceAutoCompletePrediction placeA)
        {
            var place = await googleMapsApi.GetPlaceDetails(placeA.PlaceId);
            if (place != null)
            {
                if (_isPickupFocused)
                {
                    PickupText = place.Name;
                    _originLatitud = $"{place.Latitude}";
                    _originLongitud = $"{place.Longitude}";
                    _isPickupFocused = false;
                    FocusOriginCommand.Execute(null);
                }
                else
                {
                    _destinationLatitud = $"{place.Latitude}";
                    _destinationLongitud = $"{place.Longitude}";

                    RecentPlaces.Add(placeA);

                    if (_originLatitud == _destinationLatitud && _originLongitud ==
_destinationLongitud)
                    {
                        var search = new MapView();
                        await App.Current.MainPage.DisplayAlert("", "Please Click the
Search Button", "Ok");
                    }
                    else
                    {
                        LoadRouteCommand.Execute(null);
                        await App.Current.MainPage.Navigation.PopModalAsync(false);
                        CleanFields();
                    }

                }
            }
        }

        void CleanFields()
        {
            PickupText = OriginText = string.Empty;
            ShowRecentPlaces = true;
            PlaceSelected = null;
        }

        public event PropertyChangedEventHandler PropertyChanged;

    }
}
```

Figure 36 - MainViewModel.cs

3.1.5. Views

3.1.5.1. ChangePass.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     ChangePass
    Purpose:       Change password interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.ChangePass"
            BackgroundColor="White">
   <!--Background Color-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="White"  Offset="0.7" />
           <GradientStop Color="#ff8c19" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>
       <ScrollView>
           <Grid RowSpacing="17">
               <Grid.RowDefinitions>
                   <RowDefinition Height="135"></RowDefinition>
                   <RowDefinition Height="100"></RowDefinition>
                   <RowDefinition Height="30"></RowDefinition>
                   <RowDefinition Height="50"></RowDefinition>
                   <RowDefinition Height="56"></RowDefinition>
                   <RowDefinition Height="56"></RowDefinition>
               </Grid.RowDefinitions>
               <Grid.ColumnDefinitions>
                   <ColumnDefinition Width="*"></ColumnDefinition>
                   <ColumnDefinition Width="300"></ColumnDefinition>
                   <ColumnDefinition Width="*"></ColumnDefinition>
               </Grid.ColumnDefinitions>

               <!--Change Password Label-->
               <Label       Text="Change Password"
                            HorizontalTextAlignment="Center"
                            TextColor="Black"
                            FontSize="50"
                            Margin="0, 10"
                            FontAttributes="Bold"
                            Grid.Row="0"
                            Grid.Column="1">
               </Label>

               <!--Image Import-->
               <Image       Source="xhdpi.png"
                            Grid.Row="1"
                            HeightRequest="150"
                            WidthRequest="150"
            Grid.Column="1">
</Image>

<!--Label for entering password-->
<Label      Text="Please enter your new password"
            Grid.Row="2"
            Grid.Column="1"
            FontSize="20"
            TextColor="Black"
            HorizontalTextAlignment="Center">
</Label>

<!--Entry for Password-->
<Entry      Placeholder="Password"
            PlaceholderColor="White"
            BackgroundColor="LightGray"
            TextColor="Black"
            Grid.Row="3"
            Grid.Column="1"
            x:Name="oldPass"
            IsPassword="True">
</Entry>

<!--Button to Confirm Password -->
<Button     Text="Change Password"
            Grid.Row="4"
            Grid.Column="1"
            TextColor="Black"
            BorderColor="Black"
            BorderWidth="3"
            CornerRadius="10"
            x:Name="ChangePassword"
            Clicked="ChangePassword_Clicked"
            FontAttributes="Bold"
            FontSize="20">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.6" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Back Button-->
<Button     Text="Back"
            Grid.Row="5"
            Grid.Column="1"
            TextColor="Black"
            BorderColor="Black"
            BorderWidth="3"
            CornerRadius="10"
            x:Name="BackButton"
            FontAttributes="Bold"
            FontSize="20"
```

```
                            Clicked="BackButton_Clicked"
                            ImageSource="backarrow.png"
                            ContentLayout="Left"
                            Padding="40,0,75,0">
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0,1">
                            <GradientStop Color="LightGray"  Offset="0.1" />
                            <GradientStop Color="white" Offset="1.0" />
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Figure 37 - ChangePass.xaml.

3.1.5.2 ChangePass.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   ChangePass
* Purpose:     Backend for change password functionality.
*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class ChangePass : ContentPage
   {
       IAuth auth;
       public ChangePass()
       {
           InitializeComponent();
       }

       // Change Password Event Handler
       private async void ChangePassword_Clicked(object sender, EventArgs e)
       {
           try
           {
               if(oldPass.Text == null)
               {
                   await DisplayAlert("Error", "Please enter password", "Ok");
               }
               if (oldPass.Text.Length < 8)
```

```
                {
                    await DisplayAlert("Error", "Password length must be greater than 8",
"ok");
                }
                else
                {
                    MapView.locationCount = 0;
                    var authService = DependencyService.Resolve<IAuth>();
                    await authService.ChangePassword(oldPass.Text);
                    await DisplayAlert("Success", "Password Changed", "ok");
                    await Navigation.PushModalAsync(new MainTabPage());
                }
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }

        }

        // Back Button Event Handler
        private async void BackButton_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PopModalAsync();
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }

        }
    }
}
```

Figure 38 - ChangePass.xaml.cs

3.1.5.3. FirstPageView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     FirstPageView
    Purpose:       The first Page the user can use.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.FirstPageView"
            BackgroundImageSource="b_g.png">
   <!--Content Page-->
   <ContentPage.Content>
       <ScrollView>
           <Grid RowSpacing="13">
```

```
<Grid.RowDefinitions>
    <RowDefinition Height="0"></RowDefinition>
    <RowDefinition Height="158"></RowDefinition>
    <RowDefinition Height="*"></RowDefinition>
    <RowDefinition Height="*"></RowDefinition>
    <RowDefinition Height="56"></RowDefinition>
    <RowDefinition Height="56"></RowDefinition>
    <RowDefinition Height="135"></RowDefinition>
</Grid.RowDefinitions>

<!--Header Label-->
<Label          Text="Wildfire"
                Grid.Row="1"
                FontSize="50"
                HorizontalOptions="Center"
                TextColor="Black"
                Margin="0,30"
                FontAttributes="Bold">
</Label>

<!--Message Label-->
<Label          Text="If you are a Firefighter Please create an account"
                Grid.Row="2"
                FontSize="20"
                HorizontalOptions="CenterAndExpand"
                TextColor="White"
                Margin="0,30"
                FontAttributes="Bold"
                WidthRequest="220"
                LineBreakMode="CharacterWrap"
                HorizontalTextAlignment="Center">
</Label>

<!--Login Button-->
<Button         Text="Login"
                Grid.Row="4"
                TextColor="Black"
                HorizontalOptions="Center"
                VerticalOptions="EndAndExpand"
                BorderWidth="3"
                BorderColor="Black"
                CornerRadius="10"
                FontSize="18"
                FontAttributes="Bold"
```

```
            BackgroundColor="#ff8c19"
            Clicked="LoginIn_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.7" />
            <GradientStop Color="White"  Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Create Account Button-->
<Button     Text="Create Account"
            Grid.Row="5"
            TextColor="Black"
            HorizontalOptions="Center"
            BackgroundColor="#ff8c19"
            CornerRadius="10"
            BorderWidth="3"
            FontSize="18"
            FontAttributes="Bold"
            BorderColor="Black"
            Clicked="CreateAcc_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.7" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Continue without Account Button-->
<Button     Text="Continue without Account"
            Grid.Row="6"
            TextColor="Black"
            HorizontalOptions="Center"
            VerticalOptions="EndAndExpand"
            Margin="0,20"
            CornerRadius="10"
            BorderWidth="3"
            BorderColor="Black"
            FontAttributes="Bold"
            BackgroundColor="LightGray"
            Clicked="Continue_Clicked">
    <Button.Background>
```

```
                        <LinearGradientBrush EndPoint="0,1">
                            <GradientStop Color="LightGray"  Offset="0.1" />
                            <GradientStop Color="white" Offset="1.0" />
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Figure 39 - FirstPageView.xaml

3.1.5.4. FirstPageView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   FirstPageView
* Purpose:     Backend for FirstPageView.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class FirstPageView : ContentPage
   {
       IAuth auth;
       public FirstPageView()
       {
           InitializeComponent();
           auth = DependencyService.Get<IAuth>();
       }

       protected override bool OnBackButtonPressed()
       {
           return true;
       }

       // Create Account Event Handler
       private async void CreateAcc_Clicked(object sender, EventArgs e)
       {
           var signOut = auth.SignOut();
           try
           {
               if (signOut)
               {
                   MapView.loginCount = 0;
                   await Navigation.PushModalAsync(new SignUpPageView());
               }
           }
           catch (Exception ex)
           {
               ex.Message.ToString();
           }
       }

       // Login Event Handler
       private async void LoginIn_Clicked(object sender, EventArgs e)
       {
           var signOut = auth.SignOut();
           try
           {
               if (signOut)
               {
                   await Navigation.PushModalAsync(new LoginPageView());
               }
           }
           catch (Exception ex)
           {
               ex.Message.ToString();
           }
       }

       // Continue without account Event Handler
       private async void Continue_Clicked(object sender, EventArgs e)
       {
           try
           {
               await Navigation.PushModalAsync(new MainTabPage());
           }
           catch (Exception ex)
           {
               ex.Message.ToString();
           }
       }

   }
}
```

Figure 40 - FirstPageView.xam.cs

3.1.5.5. ForgotPassword.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     ForgotPassword
    Purpose:       Forgot password interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.ForgotPassword"
```

```
         BackgroundColor="white">
<!--Background Color-->
<ContentPage.Background>
    <LinearGradientBrush EndPoint="0,1">
        <GradientStop Color="White"  Offset="0.7" />
        <GradientStop Color="#ff8c19" Offset="1.0" />
    </LinearGradientBrush>
</ContentPage.Background>

<!--Content Page-->
<ContentPage.Content>
    <ScrollView>
        <Grid RowSpacing="17">
            <Grid.RowDefinitions>
                <RowDefinition Height="135"></RowDefinition>
                <RowDefinition Height="100"></RowDefinition>
                <RowDefinition Height="30"></RowDefinition>
                <RowDefinition Height="50"></RowDefinition>
                <RowDefinition Height="56"></RowDefinition>
                <RowDefinition Height="56"></RowDefinition>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"></ColumnDefinition>
                <ColumnDefinition Width="300"></ColumnDefinition>
                <ColumnDefinition Width="*"></ColumnDefinition>
            </Grid.ColumnDefinitions>

            <!--Forgot Passoword Label-->
            <Label      Text="Forgot Password?"
                        HorizontalTextAlignment="Center"
                        TextColor="Black"
                        FontSize="50"
                        Margin="0, 10"
                        FontAttributes="Bold"
                        Grid.Row="0"
                        Grid.Column="1">
            </Label>

            <!--Image Import-->
            <Image      Source="xhdpi.png"
                        Grid.Row="1"
                        HeightRequest="150"
                        WidthRequest="150"
                        Grid.Column="1">
            </Image>

            <!--Message Label-->
            <Label      Text="Please Enter Recovery Email"
                        Grid.Row="2"
                        Grid.Column="1"
                        FontSize="20"
                        TextColor="Black"
                        HorizontalTextAlignment="Center">
            </Label>
```

```
<!--Email Entry-->
<Entry      Placeholder="Email"
            PlaceholderColor="White"
            BackgroundColor="LightGray"
            TextColor="Black"
            Grid.Row="3"
            Grid.Column="1"
            x:Name="emailPass">
</Entry>

<!--Send Email Button-->
<Button     Text="Send Email"
            Grid.Row="4"
            Grid.Column="1"
            TextColor="Black"
            BorderColor="Black"
            BorderWidth="3"
            CornerRadius="10"
            x:Name="ForgotPass"
            Clicked="ForgotPass_Clicked"
            FontAttributes="Bold"
            FontSize="20"
            ImageSource="message.png"
            ContentLayout="Right"
            Padding="70,0,30,0">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.6" />
            <GradientStop Color="White" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Back Button-->
<Button     Text="Back"
            Grid.Row="5"
            Grid.Column="1"
            BackgroundColor="LightGray"
            TextColor="Black"
            BorderColor="Black"
            BorderWidth="3"
            CornerRadius="10"
            x:Name="BackButton"
            FontAttributes="Bold"
            FontSize="20"
            Clicked="BackButton_Clicked"
            ImageSource="backarrow.png"
            ContentLayout="Left"
            Padding="40,0,75,0">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
```

```
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Figure 41 - ForgotPassword.xaml

3.1.5.6. ForgotPassword.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   ForgotPassword
* Purpose:     Backend for Forgot Password.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class ForgotPassword : ContentPage
   {
       public ForgotPassword()
       {
           InitializeComponent();
       }

       //Forgot Password Event Handler
       private async void ForgotPass_Clicked(object sender, EventArgs e)
       {
           try
           {
               if(emailPass.Text == null)
               {
                   await DisplayAlert("Error", "Please enter email", "Ok");
               }
               var authService = DependencyService.Resolve<IAuth>();
               await authService.ForgotPassword(emailPass.Text);
               await DisplayAlert("Success", "Please check email inbox", "ok");
               await Navigation.PushModalAsync(new LoginPageView());
           }
           catch(Exception ex)
           {
               ex.Message.ToString();
           }

       }
```

```
        //Back Button Event Handler
        private async void BackButton_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PushModalAsync(new LoginPageView());
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }

        }
    }
}
```

Figure 42 - ForgotPassword.xaml.cs

3.1.5.7. HomeInfoView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     HomeInfoView
    Purpose:       Home Safety info interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.HomeInfoView">
   <!--Background-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="#ff8c19"  Offset="0.1" />
           <GradientStop Color="#F7D9BB" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>

       <AbsoluteLayout>

           <!--Header-->
           <Frame      BorderColor="LightGray"
                       Padding="7"
                       HeightRequest="80"
                       WidthRequest="300"
                       AbsoluteLayout.LayoutFlags="PositionProportional"
                       AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                       BackgroundColor="LightGray"
                       HasShadow="True"
                       IsVisible="true"
                       CornerRadius="15"
                       x:Name="FrameLabel"
                       HorizontalOptions="Center">
```

```
<Frame      BorderColor="LightGray"
            Padding="7"
            HeightRequest="80"
            WidthRequest="310"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
            BackgroundColor="LightGray"
            HasShadow="True"
            CornerRadius="15"
            IsVisible="true"
            x:Name="FrameLabel1"
            HorizontalOptions="Center">
        <Frame.Background>
            <LinearGradientBrush EndPoint="0,1">
                <GradientStop Color="White"  Offset="0.9" />
                <GradientStop Color="LightBlue" Offset="1.0" />
            </LinearGradientBrush>
        </Frame.Background>

        <Label      Text="Home Fire Safety"
                    FontSize="26"
                    HorizontalOptions="Center"
                    VerticalOptions="Center"
                    TextColor="#282828"
                    FontAttributes="Bold">
        </Label>
    </Frame>
</Frame>

<!--Button 1-->
<Button     x:Name="PreventionInfo"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.25,-1,-1"
            BackgroundColor="LightBlue"
            CornerRadius="25"
            FontSize="25"
            FontAttributes="Italic"
            BorderWidth="3"
            BorderColor="Black"
            Text="Prevention"
            TextColor="Black"
            HeightRequest="110"
            WidthRequest="270"
            ImageSource="hand.png"
            Padding="0,0,30,0"
            ContentLayout="Right"
            Clicked="PreventionInfo_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue" Offset="0.3" />
            <GradientStop Color="LightSalmon" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
```

```
</Button>

<!--Button 2-->
<Button      x:Name="DetectionInfo"
             AbsoluteLayout.LayoutFlags="PositionProportional"
             AbsoluteLayout.LayoutBounds="0.5,0.45,-1,-1"
             BackgroundColor="LightBlue"
             CornerRadius="25"
             FontSize="25"
             FontAttributes="Italic"
             BorderWidth="3"
             BorderColor="Black"
             Text="Detection"
             TextColor="Black"
             HeightRequest="110"
             WidthRequest="270"
             ImageSource="detect.png"
             Padding="0,0,30,0"
             ContentLayout="Right"
             Clicked="DetectionInfo_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue" Offset="0.3" />
            <GradientStop Color="LightSalmon" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Button 3-->
<Button      x:Name="EvalutionInfo"
             AbsoluteLayout.LayoutFlags="PositionProportional"
             AbsoluteLayout.LayoutBounds="0.5,0.65,-1,-1"
             BackgroundColor="LightBlue"
             CornerRadius="25"
             FontSize="25"
             FontAttributes="Italic"
             BorderWidth="3"
             BorderColor="Black"
             Text="Evacuation"
             TextColor="Black"
             HeightRequest="110"
             WidthRequest="270"
             ImageSource="evacuation.png"
             Padding="0,0,30,0"
             ContentLayout="Right"
             Clicked="EvalutionInfo_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue" Offset="0.3" />
            <GradientStop Color="LightSalmon" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>
```

```
<Button     x:Name="BackButton"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.85,-1,-1"
            BackgroundColor="LightGray"
            CornerRadius="10"
            FontSize="20"
            FontAttributes="Bold"
            BorderWidth="3"
            BorderColor="Black"
            Text="Back"
            TextColor="Black"
            HeightRequest="56"
            WidthRequest="300"
            ImageSource="backarrow.png"
            ContentLayout="Left"
            Padding="40,0,75,0"
            Clicked="BackButton_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!-- Prevention -->
<Frame      HeightRequest="650"
            WidthRequest="310"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
            BackgroundColor="Transparent"
            x:Name="preventionInfo"
            IsVisible="false"
            HasShadow="True">

    <Frame  BackgroundColor="#f2f2f2"
            CornerRadius="15"
            HasShadow="True">

        <StackLayout>

            <Image HorizontalOptions="End"
                   HeightRequest="20"
                   WidthRequest="20"
                   Source="cancel.png">
                <Image.GestureRecognizers>
                    <TapGestureRecognizer
Tapped="PreventionRemovePopupTapped"
                                                  NumberOfTapsRequired="1">
                    </TapGestureRecognizer>
                </Image.GestureRecognizers>
            </Image>

            <Frame      BorderColor="LightGray"
```

```
                Padding="7"
                HeightRequest="80"
                WidthRequest="300"
                AbsoluteLayout.LayoutFlags="PositionProportional"
                AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                BackgroundColor="LightGray"
                HasShadow="True"
                CornerRadius="15"
                IsVisible="true"
                HorizontalOptions="Center">

            <Frame      BorderColor="LightGray"
                        Padding="7"
                        HeightRequest="80"
                        WidthRequest="310"

AbsoluteLayout.LayoutFlags="PositionProportional"
                        AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                        BackgroundColor="LightGray"
                        HasShadow="True"
                        CornerRadius="15"
                        IsVisible="true"
                        HorizontalOptions="Center">
                <Frame.Background>
                    <LinearGradientBrush EndPoint="0,1">
                        <GradientStop Color="White"  Offset="0.9" />
                        <GradientStop Color="LightBlue" Offset="1.0"
/>
                    </LinearGradientBrush>
                </Frame.Background>

                <Label      Text="Prevention"
                            FontSize="26"
                            HorizontalOptions="Center"
                            VerticalOptions="Center"
                            TextColor="#282828"
                            FontAttributes="Bold">
                </Label>
            </Frame>
        </Frame>

        <ScrollView>
            <StackLayout>

                <Label      Text="In 2019 there was 3,700 deaths from
house fires in the US."

                            TextColor="Black"
                            FontSize="18"
                            HorizontalTextAlignment="Center"
                            IsVisible="true">
                </Label>

                <BoxView    VerticalOptions="Center"
                            HorizontalOptions="Center"
```

```
                                HeightRequest="1"
                                WidthRequest="300"
                                Color="#5b5d68">
                        </BoxView>

                        <Label      Text="Follow the ( STOP ) acronym to give
yourself the best chance"
                                TextColor="Black"
                                FontSize="19"
                                FontAttributes="Bold"
                                HorizontalTextAlignment="Center">
                        </Label>

                        <BoxView    VerticalOptions="Center"
                                HorizontalOptions="Center"
                                HeightRequest="1"
                                WidthRequest="300"
                                Color="#5b5d68">
                        </BoxView>

                        <Label      Text="S  -  Smoke Alarms"
                                TextColor="DarkRed"
                                FontSize="20"
                                FontAttributes="Bold"
                                IsVisible="true"
                                x:Name="TopLabelS">
                        </Label>

                        <Label      Text="Make sure you have smoke alarms in your
home/ living space."
                                TextColor="Black"
                                FontSize="16"
                                FontAttributes="None"
                                IsVisible="true">
                        </Label>

                        <BoxView    VerticalOptions="Center"
                                HorizontalOptions="Center"
                                HeightRequest="1"
                                WidthRequest="300"
                                Color="#5b5d68">
                        </BoxView>

                        <Label      Text="T  -  Test your Smoke Alarms"
                                TextColor="DarkRed"
                                FontSize="20"
                                FontAttributes="Bold"
                                IsVisible="true">
                        </Label>

                        <Label      Text="Test your alarms frequently."
                                TextColor="Black"
                                FontSize="16"
                                FontAttributes="None"
```

```
                                    IsVisible="true">
                            </Label>

                            <BoxView    VerticalOptions="Center"
                                        HorizontalOptions="Center"
                                        HeightRequest="1"
                                        WidthRequest="300"
                                        Color="#5b5d68">
                            </BoxView>

                            <Label      Text="O  -  Obvious Dangers"
                                        TextColor="DarkRed"
                                        FontSize="20"
                                        FontAttributes="Bold"
                                        IsVisible="true">
                            </Label>

                            <Label      Text="Keep an eye on obvious danger points
e.g. Candles and Chimneys ."

                                        TextColor="Black"
                                        FontSize="16"
                                        FontAttributes="None"
                                        IsVisible="true">
                            </Label>

                            <BoxView    VerticalOptions="Center"
                                        HorizontalOptions="Center"
                                        HeightRequest="1"
                                        WidthRequest="300"
                                        Color="#5b5d68">
                            </BoxView>

                            <Label      Text="P  -  Have a Plan"
                                        TextColor="DarkRed"
                                        FontSize="20"
                                        FontAttributes="Bold"
                                        IsVisible="true">
                            </Label>

                            <Label      Text="Have a escape route for all floor plans
and always know house keys are stored."

                                        TextColor="Black"
                                        FontSize="16"
                                        FontAttributes="None"
                                        IsVisible="true">
                            </Label>

                        </StackLayout>
                    </ScrollView>
                </StackLayout>
            </Frame>
        </Frame>

        <!-- Detection -->
```

```
<Frame      HeightRequest="600"
            WidthRequest="310"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
            BackgroundColor="Transparent"
            x:Name="detectionInfo"
            IsVisible="false"
            HasShadow="True">

    <Frame  BackgroundColor="#f2f2f2"
            CornerRadius="15"
            HasShadow="True">

        <StackLayout>

            <Image HorizontalOptions="End"
                   HeightRequest="20"
                   WidthRequest="20"
                   Source="cancel.png">
                <Image.GestureRecognizers>
                    <TapGestureRecognizer Tapped="DetectionRemovePopupTapped"
                                          NumberOfTapsRequired="1">
                    </TapGestureRecognizer>
                </Image.GestureRecognizers>
            </Image>

            <Frame      BorderColor="LightGray"
                        Padding="7"
                        HeightRequest="80"
                        WidthRequest="300"
                        AbsoluteLayout.LayoutFlags="PositionProportional"
                        AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                        BackgroundColor="LightGray"
                        HasShadow="True"
                        CornerRadius="15"
                        IsVisible="true"
                        HorizontalOptions="Center">

                <Frame      BorderColor="LightGray"
                            Padding="7"
                            HeightRequest="80"
                            WidthRequest="310"
                            AbsoluteLayout.LayoutFlags="PositionProportional"
                            AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                            BackgroundColor="LightGray"
                            HasShadow="True"
                            CornerRadius="15"
                            IsVisible="true"
                            HorizontalOptions="Center">
                            <Frame.Background>
                                <LinearGradientBrush EndPoint="0,1">
                                    <GradientStop Color="White"  Offset="0.9"
/>

                                    <GradientStop Color="LightBlue"
```

```
Offset="1.0" />
                                            </LinearGradientBrush>
                                        </Frame.Background>

                                    <Label      Text="Detection"
                                                FontSize="26"
                                                HorizontalOptions="Center"
                                                VerticalOptions="Center"
                                                TextColor="#282828"
                                                FontAttributes="Bold">
                                    </Label>
                            </Frame>
                        </Frame>

                        <ScrollView>
                            <StackLayout>

                                <Label      Text="A fire will not wake you up while you
are sleeping."

                                            TextColor="Black"
                                            FontSize="18"
                                            HorizontalTextAlignment="Center"
                                            IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                            HorizontalOptions="Center"
                                            HeightRequest="1"
                                            WidthRequest="300"
                                            Color="#5b5d68">
                                </BoxView>

                                <Label      Text="It will put you further to sleep!"
                                            TextColor="Black"
                                            FontSize="18"
                                            HorizontalTextAlignment="Center"
                                            FontAttributes="Bold"
                                            IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                            HorizontalOptions="Center"
                                            HeightRequest="1"
                                            WidthRequest="300"
                                            Color="#5b5d68">
                                </BoxView>

                                <Label      Text="The following preventative measures are
recommended"

                                            TextColor="Black"
                                            FontSize="18"
                                            HorizontalTextAlignment="Center"
                                            IsVisible="true">
                                </Label>
```

```
<BoxView    VerticalOptions="Center"
            HorizontalOptions="Center"
            HeightRequest="1"
            WidthRequest="300"
            Color="#5b5d68">
</BoxView>

<Label      Text=" - Smoke Alarms"
            TextColor="DarkRed"
            FontSize="20"
            HorizontalTextAlignment="Start"
            FontAttributes="Bold"
            IsVisible="true">
</Label>

<Label      Text="Have one smoke alarms per floor. "
            TextColor="Black"
            FontSize="18"
            HorizontalTextAlignment="Center"
            IsVisible="true">
</Label>

<BoxView    VerticalOptions="Center"
            HorizontalOptions="Center"
            HeightRequest="1"
            WidthRequest="300"
            Color="#5b5d68">

</BoxView>

<Label      Text=" - Change Batteries"
            TextColor="DarkRed"
            FontSize="20"
            HorizontalTextAlignment="Start"
            FontAttributes="Bold"
            IsVisible="true">
</Label>

<Label      Text="Change Batteries once a year. "
            TextColor="Black"
            FontSize="18"
            HorizontalTextAlignment="Start"
            IsVisible="true">
</Label>

<BoxView    VerticalOptions="Center"
            HorizontalOptions="Center"
            HeightRequest="1"
            WidthRequest="300"
            Color="#5b5d68">
</BoxView>

<Label      Text="- Test"
```

```
                                           TextColor="DarkRed"
                                           FontSize="20"
                                           HorizontalTextAlignment="Start"
                                           FontAttributes="Bold"
                                           IsVisible="true">
                               </Label>

                               <Label      Text="Test smoke alarm once a week. "
                                           TextColor="Black"
                                           FontSize="18"
                                           HorizontalTextAlignment="Start"
                                           IsVisible="true">
                               </Label>

                           </StackLayout>
                       </ScrollView>
                   </StackLayout>
               </Frame>
           </Frame>

           <!-- Evaluation -->
           <Frame      HeightRequest="676"
                       WidthRequest="310"
                       AbsoluteLayout.LayoutFlags="PositionProportional"
                       AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
                       BackgroundColor="Transparent"
                       x:Name="evaluationInfo"
                       IsVisible="false"
                       HasShadow="True">

               <Frame      BackgroundColor="#f2f2f2"
                           CornerRadius="15"
                           HasShadow="True">

                   <StackLayout>

                       <Image      HorizontalOptions="End"
                                   HeightRequest="20"
                                   WidthRequest="20"
                                   Source="cancel.png">
                           <Image.GestureRecognizers>
                               <TapGestureRecognizer
Tapped="EvaluationRemovePopupTapped"
                                                     NumberOfTapsRequired="1">
                               </TapGestureRecognizer>
                           </Image.GestureRecognizers>
                       </Image>

                       <Frame      BorderColor="LightGray"
                                   Padding="7"
                                   HeightRequest="80"
                                   WidthRequest="300"
                                   AbsoluteLayout.LayoutFlags="PositionProportional"
                                   AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                                   BackgroundColor="LightGray"
                                   HasShadow="True"
                                   CornerRadius="15"
                                   IsVisible="true"
                                   HorizontalOptions="Center">

                           <Frame      BorderColor="LightGray"
                                           Padding="7"
                                           HeightRequest="80"
                                           WidthRequest="310"

AbsoluteLayout.LayoutFlags="PositionProportional"
                                           AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                                           BackgroundColor="LightGray"
                                           HasShadow="True"
                                           CornerRadius="15"
                                           IsVisible="true"
                                           HorizontalOptions="Center">
                               <Frame.Background>
                                   <LinearGradientBrush EndPoint="0,1">
                                       <GradientStop Color="White"  Offset="0.9" />
                                       <GradientStop Color="LightBlue" Offset="1.0" />
                                   </LinearGradientBrush>
                               </Frame.Background>

                               <Label      Text="Evacuation"
                                               FontSize="26"
                                               HorizontalOptions="Center"
                                               VerticalOptions="Center"
                                               TextColor="#282828"
                                               FontAttributes="Bold">
                               </Label>

                           </Frame>
                       </Frame>

                       <ScrollView>
                           <StackLayout>

                               <Label      Text="Knowing what to do when a fire alarms
sounds is as important as having one."
                                           TextColor="Black"
                                           FontSize="18"
                                           HorizontalTextAlignment="Center"
                                           IsVisible="true">
                               </Label>

                               <BoxView    VerticalOptions="Center"
                                           HorizontalOptions="Center"
                                           HeightRequest="1"
                                           WidthRequest="300"
                                           Color="#5b5d68">
                               </BoxView>
```

```
                                <Label      Text="Follow the ( RACE ) acronym to give
yourself the best chance."

                                            TextColor="Black"
                                            FontSize="18"
                                            FontAttributes="Bold"
                                            HorizontalTextAlignment="Center"
                                            IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                            HorizontalOptions="Center"
                                            HeightRequest="1"
                                            WidthRequest="300"
                                            Color="#5b5d68">
                                </BoxView>

                                <Label      Text="R - Remove"
                                            TextColor="DarkRed"
                                            FontSize="20"
                                            HorizontalTextAlignment="Start"
                                            FontAttributes="Bold"
                                            IsVisible="true">
                                </Label>

                                <Label      Text="Remove all personal in the building to
a safe point."

                                            TextColor="Black"
                                            FontSize="18"
                                            HorizontalTextAlignment="Start"
                                            IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                            HorizontalOptions="Center"
                                            HeightRequest="1"
                                            WidthRequest="300"
                                            Color="#5b5d68">
                                </BoxView>

                                <Label      Text="A - Alert"
                                            TextColor="DarkRed"
                                            FontSize="20"
                                            HorizontalTextAlignment="Start"
                                            FontAttributes="Bold"
                                            IsVisible="true">
                                </Label>

                                <Label      Text="Alert the emergeny service as soon as
possible."

                                            TextColor="Black"
                                            FontSize="18"
                                            HorizontalTextAlignment="Start"
                                            IsVisible="true">
                                </Label>
```

```
                              <BoxView    VerticalOptions="Center"
                                          HorizontalOptions="Center"
                                          HeightRequest="1"
                                          WidthRequest="300"
                                          Color="#5b5d68">
                              </BoxView>

                              <Label      Text="C - Contain"
                                          TextColor="DarkRed"
                                          FontSize="20"
                                          HorizontalTextAlignment="Start"
                                          FontAttributes="Bold"
                                          IsVisible="true">
                              </Label>

                              <Label      Text="Contain the fire by closing doors and 
trying to stop the fire from spreading."
                                          TextColor="Black"
                                          FontSize="18"
                                          HorizontalTextAlignment="Start"
                                          IsVisible="true">
                              </Label>

                              <BoxView    VerticalOptions="Center"
                                          HorizontalOptions="Center"
                                          HeightRequest="1"
                                          WidthRequest="300"
                                          Color="#5b5d68">
                              </BoxView>

                              <Label      Text="E - Extinguish"
                                          TextColor="DarkRed"
                                          FontSize="20"
                                          HorizontalTextAlignment="Start"
                                          FontAttributes="Bold"
                                          IsVisible="true">
                              </Label>

                              <Label      Text="If fire extinguishers are available, 
Safely try to contain the fire."
                                          TextColor="Black"
                                          FontSize="18"
                                          HorizontalTextAlignment="Start"
                                          IsVisible="true">
                              </Label>

                          </StackLayout>
                      </ScrollView>
                  </StackLayout>
              </Frame>
          </Frame>
      </AbsoluteLayout>
  </ContentPage.Content>
```

```
</ContentPage>
```

Figure 43 - HomeInfoView.xaml

3.1.5.8. HomeInfoView.xaml.cs

```
/* Author:      Jack McNally
 * Page Name:   HomeInfoView
 * Purpose:     Backend for Home safety info.
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class HomeInfoView : ContentPage
   {
       public HomeInfoView()
       {
           InitializeComponent();
       }

       // Prevention Button Handler
       private void PreventionInfo_Clicked(object sender, EventArgs e)
       {
           try
           {
               FrameLabel.IsVisible = false;
               FrameLabel1.IsVisible = false;
               preventionInfo.IsVisible = true;
               PreventionInfo.IsVisible = false;
           }
           catch(Exception ex)
           {
               ex.Message.ToString();
           }
       }

       // Prevention Button Remove Handler
       void PreventionRemovePopupTapped(object sender, EventArgs e)
       {
           FrameLabel.IsVisible = true;
           FrameLabel1.IsVisible = true;
           preventionInfo.IsVisible = false;
           PreventionInfo.IsVisible = true;
       }

       // Detection Button Handler
       private void DetectionInfo_Clicked(object sender, EventArgs e)
```

```
{
    try
    {
        FrameLabel.IsVisible = false;
        FrameLabel1.IsVisible = false;
        detectionInfo.IsVisible = true;
        DetectionInfo.IsVisible = false;
    }
    catch (Exception ex)
    {
        ex.Message.ToString();
    }
}

// Detection Button Remove Handler
void DetectionRemovePopupTapped(object sender, EventArgs e)
{
    FrameLabel.IsVisible = true;
    FrameLabel1.IsVisible = true;
    detectionInfo.IsVisible = false;
    DetectionInfo.IsVisible = true;
}

// Evacuation Button Handler
private void EvalutionInfo_Clicked(object sender, EventArgs e)
{
    try
    {
        FrameLabel.IsVisible = false;
        FrameLabel1.IsVisible = false;
        evaluationInfo.IsVisible = true;
        EvalutionInfo.IsVisible = false;
    }
    catch (Exception ex)
    {
        ex.Message.ToString();
    }
}

// Evacuation Button Remove Handler
void EvaluationRemovePopupTapped(object sender, EventArgs e)
{
    FrameLabel.IsVisible = true;
    FrameLabel1.IsVisible = true;
    evaluationInfo.IsVisible = false;
    EvalutionInfo.IsVisible = true;
}
 // Back Button Handler
private async void BackButton_Clicked(object sender, EventArgs e)
{
    try
    {
        await Navigation.PopModalAsync();
    }
```

```
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }
    }
}
```

Figure 44 - HomeInfoView.xaml.cs

3.1.5.9. InfoView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
     Page Name:     InfoView
     Purpose:       Information view interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Wildfire.Views.InfoView"
             Title="Info Centre"
             >
   <!--Background Color-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="#ff8c19"  Offset="0.1" />
           <GradientStop Color="#F7D9BB" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>

       <ScrollView>
           <Grid>
               <Grid.RowDefinitions>
                   <RowDefinition Height="*"></RowDefinition>
                   <RowDefinition Height="130"></RowDefinition>
                   <RowDefinition Height="*"></RowDefinition>
                   <RowDefinition Height="130"></RowDefinition>
                   <RowDefinition Height="*"></RowDefinition>
                   <RowDefinition Height="130"></RowDefinition>
                   <RowDefinition Height="*"></RowDefinition>
                   <RowDefinition Height="130"></RowDefinition>
                   <RowDefinition Height="*"></RowDefinition>
               </Grid.RowDefinitions>
               <Grid.ColumnDefinitions>
                   <ColumnDefinition Width="*"/>
                   <ColumnDefinition Width="300"/>
                   <ColumnDefinition Width="*"/>
               </Grid.ColumnDefinitions>

               <!--Home Button-->
               <Button x:Name="HomeInfo"
                       Grid.Row="1"
```

```
                Grid.Column="1"
                CornerRadius="25"
                FontSize="25"
                FontAttributes="Italic"
                BorderWidth="3"
                BorderColor="Black"
                TextColor="Black"
                BackgroundColor="LightBlue"
                Padding="0,0,40,0"
                Text="Home"
                ContentLayout="Right"
                ImageSource="home.png"
                Clicked="HomeInfo_Clicked">
            <Button.Background>
                <LinearGradientBrush EndPoint="1,0">
                    <GradientStop Color="LightBlue" Offset="0.3" />
                    <GradientStop Color="LightSalmon" Offset="1.0" />
                </LinearGradientBrush>
            </Button.Background>
        </Button>

        <!--Wildland Button-->
        <Button x:Name="WildlandInfo"
                Grid.Row="3"
                Grid.Column="1"
                CornerRadius="25"
                Padding="0,0,40,0"
                FontSize="25"
                FontAttributes="Italic"
                BorderWidth="3"
                BorderColor="Black"
                Text="Wildland"
                TextColor="Black"
                ContentLayout="Right"
                ImageSource="tree.png"
                Clicked="WildlandInfo_Clicked">
            <Button.Background>
                <LinearGradientBrush EndPoint="1,0">
                    <GradientStop Color="LightBlue" Offset="0.3" />
                    <GradientStop Color="LightSalmon" Offset="1.0" />
                </LinearGradientBrush>
            </Button.Background>
        </Button>

        <!--Local Button -->
        <Button x:Name="LocalInfo"
                Grid.Row="5"
                Grid.Column="1"
                BackgroundColor="LightBlue"
                CornerRadius="25"
                FontSize="25"
                FontAttributes="Italic"
                BorderWidth="3"
                BorderColor="Black"
```

```
                        Text="Local Info"
                        Padding="0,0,40,0"
                        TextColor="Black"
                        ContentLayout="Right"
                        ImageSource="local.png"
                        Clicked="LocalInfo_Clicked">
                    <Button.Background>
                        <LinearGradientBrush EndPoint="1,0">
                            <GradientStop Color="LightBlue" Offset="0.3" />
                            <GradientStop Color="LightSalmon" Offset="1.0" />
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>

                <!--Spacing Line-->
                <BoxView    VerticalOptions="Center"
                            HorizontalOptions="Center"
                            HeightRequest="1"
                            Grid.Row="6"
                            Grid.Column="1"
                            WidthRequest="300"
                            Color="#5b5d68">
                </BoxView>

                <!--Firefghter Button-->
                <Button x:Name="ProfessionalInfo"
                        Grid.Row="7" Grid.Column="1"
                        BackgroundColor="#ff8c19"
                        CornerRadius="25"
                        FontSize="25"
                        FontAttributes="Bold"
                        BorderWidth="3"
                        BorderColor="Black"
                        Text="Firefighters"
                        TextColor="Black"
                        Clicked="ProfessionalInfo_Clicked"   >
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0,1">
                            <GradientStop Color="#ff8c19" Offset="0.3" />
                            <GradientStop Color="LightGray" Offset="1.0" />
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Figure 45 - InfoView.xaml

3.1.5.10. InfoView.xaml.cs

```
/* Author:      Jack McNally
 * Page Name:   InfoView
 * Purpose:     Backend for Information View.
```

```
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class InfoView : ContentPage
   {

       public InfoView()
       {
           InitializeComponent();

       }

       protected override bool OnBackButtonPressed()
       {
           return true;
       }

       // Local Info Button Handler
       private async void LocalInfo_Clicked(object sender, EventArgs e)
       {
           await Navigation.PushModalAsync(new LocalInfoView());
       }

       // Firefighter Info Button Handler
       private async void ProfessionalInfo_Clicked(object sender, EventArgs e)
       {
           try
           {
               if(LoginPageView.token != null)
               {
                   await Navigation.PushModalAsync(new ProFireInfoView());
               }
               else
               {
                   await DisplayAlert("Error", "Only Firefighters can access this Page", 
"ok");
               }
           }
           catch(Exception ex)
           {
               ex.Message.ToString();
           }
       }
```

```
        // Home Info Button Handler
        private async void HomeInfo_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PushModalAsync(new HomeInfoView());
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Wildland Info Button Handler
        private async void WildlandInfo_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PushModalAsync(new WildlandInfoView());
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }
    }
}
```

Figure 46 - InfoView.xaml.cs

3.1.5.11. LocalInfoView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     LocalInfoView
    Purpose:       Local Info interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.LocalInfoView">

   <!--Background Color-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="White"  Offset="0.7" />
           <GradientStop Color="#ff8c19" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>

       <ScrollView>
           <Grid RowSpacing="10">
               <Grid.RowDefinitions>
```

```
        <RowDefinition Height="10"></RowDefinition>
        <RowDefinition Height="80"></RowDefinition>
        <RowDefinition Height="0"></RowDefinition>
        <RowDefinition Height="30"></RowDefinition>
        <RowDefinition Height="30"></RowDefinition>
        <RowDefinition Height="0"></RowDefinition>
        <RowDefinition Height="50"></RowDefinition>
        <RowDefinition Height="0"></RowDefinition>
        <RowDefinition Height="40"></RowDefinition>
        <RowDefinition Height="10"></RowDefinition>
        <RowDefinition Height="56"></RowDefinition>
        <RowDefinition Height="0"></RowDefinition>
        <RowDefinition Height="56"></RowDefinition>

    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="300"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>

    <!--Header-->
    <Frame       BorderColor="LightGray"
                 Padding="7"
                 HeightRequest="80"
                 WidthRequest="300"
                 Grid.Row="1"
                 Grid.Column="1"
                 BackgroundColor="LightGray"
                 HasShadow="True"
                 IsVisible="true"
                 CornerRadius="15"
                 x:Name="FrameLabel"
                 HorizontalOptions="Center">

        <Frame       BorderColor="LightGray"
                     Padding="7"
                     HeightRequest="80"
                     WidthRequest="310"
                     Grid.Row="1"
                     Grid.Column="1"
                     BackgroundColor="LightGray"
                     HasShadow="True"
                     CornerRadius="15"
                     IsVisible="true"
                     x:Name="FrameLabel1"
                     HorizontalOptions="Center">
                 <Frame.Background>
                     <LinearGradientBrush EndPoint="0,1">
                         <GradientStop Color="White"  Offset="0.9" />
                         <GradientStop Color="LightBlue" Offset="1.0" />
                     </LinearGradientBrush>
                 </Frame.Background>
        <Label      Text="Local Information"
                    FontSize="26"
                    HorizontalOptions="Center"
                    VerticalOptions="Center"
                    TextColor="#282828"
                    FontAttributes="Bold">
        </Label>
    </Frame>
</Frame>

<!--Current Location -->
<Label      Grid.Row="3"
            Grid.Column="1"
            Text="Current Location"
            TextColor="Black"
            FontSize="20"
            FontAttributes="Bold"
            IsVisible="true"
            x:Name="TopLabel">
</Label>

<!--Location Binding-->
<Label      Grid.Row="4"
            Grid.Column="1"
            BackgroundColor="#2daff0"
            TextColor="Black"
            FontSize="18"
            VerticalTextAlignment="Center"
            Padding="10,0,0,0"
            WidthRequest="200"
            x:Name="InfoLocation"
            IsVisible="true">
    <Label.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue"  Offset="0.5" />
            <GradientStop Color="#2daff0" Offset="1.0" />
        </LinearGradientBrush>
    </Label.Background>
</Label>

<!--Local Number-->
<Label      Grid.Row="6"
            Grid.Column="1"
            Text="Local Emergency Contact Information"
            TextColor="Black"
            FontAttributes="Bold"
            FontSize="20"
            x:Name="EmerLabel"
            IsVisible="true"
            HorizontalTextAlignment="Start">
</Label>

<!--Location Number Binding-->
<Label      Grid.Row="8"
```

```
            Grid.Column="1"
            BackgroundColor="#2daff0"
            TextColor="Black"
            VerticalTextAlignment="Center"
            Padding="10,0,0,0"
            FontSize="18"
            x:Name="EmerNum"
            IsVisible="true">
    <Label.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue"  Offset="0.3" />
            <GradientStop Color="#2daff0" Offset="1.0" />
        </LinearGradientBrush>
    </Label.Background>
</Label>

<!--Dialer Button-->
<Button     Grid.Row="10"
            Grid.Column="1"
            BorderColor="Black"
            BorderWidth="3"
            FontSize="20"
            BackgroundColor="#ff8c19"
            CornerRadius="10"
            Text="Dial Emergency Number"
            TextColor="Black"
            FontAttributes="Bold"
            x:Name="DialerButton"
            Clicked="DialerButton_Clicked"
            IsVisible="true">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.7" />
            <GradientStop Color="White" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Back Button-->
<Button     Grid.Row="12"
            Grid.Column="1"
            BorderColor="Black"
            BorderWidth="3"
            FontSize="20"
            BackgroundColor="LightGray"
            CornerRadius="10"
            Text="Back"
            TextColor="Black"
            FontAttributes="Bold"
            x:Name="BackButton"
            Clicked="BackButton_Clicked"
            IsVisible="true"
            ImageSource="backarrow.png"
            ContentLayout="Left"
```

```
                    Padding="40,0,75,0">
                <Button.Background>
                    <LinearGradientBrush EndPoint="0,1">
                        <GradientStop Color="LightGray"  Offset="0.1" />
                        <GradientStop Color="white" Offset="1.0" />
                    </LinearGradientBrush>
                </Button.Background>
            </Button>

            <!--Activity Indicator-->
            <ActivityIndicator x:Name="overlay"
                               IsRunning="True"
                               IsVisible="True"
                               Color="Black"
                               BackgroundColor="Transparent"
                               AbsoluteLayout.LayoutFlags="PositionProportional"
                               AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
                               Grid.Column="1"
                               Grid.Row="6">
            </ActivityIndicator>

            <!--Indicator Label-->
            <Label  x:Name="loading"
                    Text="Loading Local Information..."
                    TextColor="Black"
                    FontSize="20"
                    IsVisible="false"
                    AbsoluteLayout.LayoutFlags="PositionProportional"
                    AbsoluteLayout.LayoutBounds="0.5,0.6,-1,-1"
                    HorizontalTextAlignment="Center"
                    Grid.Column="1"
                    Grid.Row="8">
            </Label>
        </Grid>
    </ScrollView>
  </ContentPage.Content>
</ContentPage>
```

Figure 47 - LocalInfoView.xaml

3.1.5.12. LocalInfoView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   LocalInfoView
* Purpose:     Backend for Local Info.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Net.Http;
using System.Collections.ObjectModel;
using Newtonsoft.Json;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]

   public partial class LocalInfoView : ContentPage
   {
       public LocalInfoView()
       {
           InitializeComponent();

       }

       /// <summary>
       /// Loading Logic
       /// </summary>
       protected async override void OnAppearing()
       {

           base.OnAppearing();

           overlay.IsVisible = true;
           loading.IsVisible = true;
           FrameLabel.IsVisible = false;
           TopLabel.IsVisible = false;
           InfoLocation.IsVisible = false;
           EmerLabel.IsVisible = false;
           EmerNum.IsVisible = false;
           DialerButton.IsVisible = false;
           BackButton.IsVisible = false;
           await LoadLocation();
           overlay.IsVisible = false;
           loading.IsVisible = false;
           FrameLabel.IsVisible = true;
           TopLabel.IsVisible = true;
           InfoLocation.IsVisible = true;
           EmerLabel.IsVisible = true;
           EmerNum.IsVisible = true;
           DialerButton.IsVisible = true;
```

```
        BackButton.IsVisible = true;
    }

    /// <summary>
    /// Load the current location of the user
    /// </summary>
    /// <returns></returns>
    async Task LoadLocation()
    {

        var location = await Geolocation.GetLocationAsync();
        if (location != null)
        {
            var plLat = location.Latitude;
            var plLong = location.Longitude;
            var placemark1 = await Geocoding.GetPlacemarksAsync(plLat, plLong);
            var placemarkDetails1 = placemark1?.FirstOrDefault();
            string locality1 = placemarkDetails1.AdminArea;
            string areaCode1 = placemarkDetails1.CountryCode;
            string Place1 = locality1 + ", " + areaCode1;

            InfoLocation.Text = Place1.ToString();

            if (areaCode1 == "IE")
            {
                var num = "999";
                EmerNum.Text = num;
            }
            else if (areaCode1 == "GB")
            {
                var num = "999";
                EmerNum.Text = num;

            }
            else
            {
                var num = "Doesn't Exist";
                EmerNum.Text = num;
            }
        }


    }

    // Back Button Handler
    private async void BackButton_Clicked(object sender, EventArgs e)
    {
        try
        {
            await Navigation.PopModalAsync();
        }
        catch(Exception ex)
        {
            ex.Message.ToString();
```

```
                }
            }

            // Dialer Button Handler
            private void DialerButton_Clicked(object sender, EventArgs e)
            {
                var emergencyNum = EmerNum.Text;
                if (emergencyNum != null)
                {
                    try
                    {
                        PhoneDialer.Open(emergencyNum);
                    }
                    catch (ArgumentNullException anEx)
                    {
                        anEx.Message.ToString();
                    }
                    catch (FeatureNotSupportedException ex)
                    {
                        ex.Message.ToString();
                    }
                    catch (Exception ex)
                    {
                        ex.Message.ToString();
                    }
                }
                else
                {
                    DisplayAlert("Error", "No number", "ok");
                }
            }
        }
}
```

Figure 48 - LocalInfoView.xaml.cs

3.1.5.13.LoginPageView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     LoginPageView
    Purpose:       Login Page interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.LoginPageView">

   <!--Background-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="White"  Offset="0.8" />
           <GradientStop Color="#ff8c19" Offset="1.0" />
```

```
        </LinearGradientBrush>
    </ContentPage.Background>

    <!--Content Page-->
    <ContentPage.Content>
        <ScrollView>
            <Grid RowSpacing="17">
                <Grid.RowDefinitions>
                    <RowDefinition Height="90"></RowDefinition>
                    <RowDefinition Height="105"></RowDefinition>
                    <RowDefinition Height="0"></RowDefinition>
                    <RowDefinition Height="50"></RowDefinition>
                    <RowDefinition Height="50"></RowDefinition>
                    <RowDefinition Height="56"></RowDefinition>
                    <RowDefinition Height="56"></RowDefinition>
                    <RowDefinition Height="40"></RowDefinition>
                    <RowDefinition Height="56"></RowDefinition>
                </Grid.RowDefinitions>

                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="300"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <!--Login Label-->
                <Label       Text="Login"
                             Grid.Row="0"
                             FontSize="50"
                             HorizontalOptions="Center"
                             TextColor="Black"
                             Margin="0,10"
                             FontAttributes="Bold"
                             Grid.Column="1">
                </Label>

                <!--Image Import-->
                <Image       Source="xhdpi.png"
                             Grid.Row="1"
                             HeightRequest="150"
                             WidthRequest="150"
                             Grid.Column="1">
                </Image>

                <!--Email Entry-->
                <Entry       Grid.Row="3"
                             Placeholder="E-mail"
                             Keyboard="Email"
                             x:Name="EmailInput"
                             BackgroundColor="LightGray"
                             WidthRequest="200"
                             PlaceholderColor="DarkGray"
                             TextColor="black"
            Grid.Column="1">
</Entry>

<!-- Password Entry-->
<Entry      Grid.Row="4"
            Placeholder="Password"
            IsPassword="True"
            x:Name="PasswordInput"
            BackgroundColor="LightGray"
            PlaceholderColor="DarkGray"
            TextColor="black"
            Grid.Column="1">
</Entry>

<!--Login Button-->
<Button     Grid.Row="5"
            Grid.Column="1"
            WidthRequest="300"
            Text="Login"
            Clicked="Login_Clicked"
            FontAttributes="Bold"
            BorderColor="Black"
            TextColor="Black"
            HorizontalOptions="Center"
            BackgroundColor="#ff8c19"
            CornerRadius="10"
            BorderWidth="3"
            FontSize="20">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.7" />
            <GradientStop Color="White" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Create Account Button-->
<Button     Grid.Row="6"
            Text="Create Account"
            TextColor="Black"
            HorizontalOptions="Center"
            BackgroundColor="LightGray"
            BorderColor="Black"
            BorderWidth="3"
            CornerRadius="10"
            Clicked="SignUp_Clicked"
            FontAttributes="Bold"
            WidthRequest="195"
            FontSize="20"
            Grid.Column="1">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Forgot Button-->
<Button     Grid.Row="7"
            Text="Forgot Password?"
            HorizontalOptions="Center"
            TextColor="Black"
            BackgroundColor="White"
            FontSize="12"
            FontAttributes="Bold"
            Margin="10,0"
            x:Name="Forgot_Button"
            Clicked="Forgot_Button_Clicked"
            Grid.Column="1"
            BorderColor="Black"
            WidthRequest="195"
            BorderWidth="3"
            CornerRadius="10">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">

            <GradientStop Color="LightBlue"  Offset="0.9" />
            <GradientStop Color="White" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Back Button-->
<Button     Grid.Row="8"
            Text="Back"
            Clicked="Back_Clicked"
            HorizontalOptions="Center"
            BackgroundColor="LightGray"
            TextColor="black"
            BorderColor="Black"
            CornerRadius="10"
            BorderWidth="3"
            FontSize="20"
            FontAttributes="Bold"
            Grid.Column="1"
            WidthRequest="300"
            ImageSource="backarrow.png"
            ContentLayout="Left"
            Padding="40,0,75,0">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
```

```
                </Button>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Figure 49 - LoginPageView.xaml

3.1.5.14. LoginPageView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   LoginPageView
* Purpose:     Backend for Login Page.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class LoginPageView : ContentPage
   {
       IAuth auth;
       public static string token;
       public LoginPageView()
       {
           InitializeComponent();
           auth = DependencyService.Get<IAuth>();
           MapView.notificationCount = 0;
           MapView.loginCount = 0;
       }

       // Login Button Handler
       private async void Login_Clicked(object sender, EventArgs e)
       {
           if(EmailInput.Text == null)
           {
               await DisplayAlert("Error", "Please enter email", "Ok");
               return;
           }
           if (PasswordInput.Text == null)
           {
               await DisplayAlert("Error", "Please enter password", "Ok");
               return;
```

```
            }
            token = await auth.LoginWithEmailAndPassword(EmailInput.Text,
PasswordInput.Text);

            try
            {

                if (token != string.Empty)
                {
                    MapView.loginCount = 0;
                    await Navigation.PushModalAsync(new MainTabPage());
                }
                else
                {
                    await DisplayAlert("Auth Failed", "Email & Password Incorrect",
"Ok");
                }
            }
            catch (Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Sign Up Button Handler
        private async void SignUp_Clicked(object sender, EventArgs e)
        {
            var signOut = auth.SignOut();

            try
            {
                if (signOut)
                {
                    await Navigation.PushModalAsync(new SignUpPageView());
                }
            }
            catch (Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Back Button Handler
        private async void Back_Clicked(object sender, EventArgs e)
        {
            try
            {
                MapView.loginCount = 0;
                await Navigation.PushModalAsync(new FirstPageView());
            }
            catch (Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Forgot Button Handler
        private async void Forgot_Button_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PushModalAsync(new ForgotPassword());
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }
    }
}
```

Figure 50 - LoginPageView.xaml.cs

3.1.5.15. MapView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     MapView
    Purpose:       Map View interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.MapView"

xmlns:maps="clr-namespace:Xamarin.Forms.GoogleMaps;assembly=Xamarin.Forms.GoogleMaps
"
            Title="Map"
            xmlns:vm="clr-namespace:Wildfire.ViewModels"
            FocusOriginCommand="{Binding FocusOriginCommand}">

   <!--Background-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="#ff8c19"  Offset="0.1" />
           <GradientStop Color="White" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Binding-->
   <ContentPage.BindingContext>
       <vm:MainViewModel/>
   </ContentPage.BindingContext>

   <!--Absolute Layout Page-->
   <AbsoluteLayout     HorizontalOptions="FillAndExpand"
                       VerticalOptions="FillAndExpand">

       <!--Map -->
       <maps:Map    x:Name="map"
                    AbsoluteLayout.LayoutBounds="0,0,1,1"
                    AbsoluteLayout.LayoutFlags="All"
                    HorizontalOptions="FillAndExpand"
                    VerticalOptions="FillAndExpand"
                    maps:MapLongClicked="map_MapClicked"
                    PinClicked="map_PinClicked"
                    IsVisible="true">
       </maps:Map>

       <!--Activity Indicator-->
       <ActivityIndicator  x:Name="overlay"
                           IsRunning="True"
                           IsVisible="True"
                           Color="Black"
                           BackgroundColor="Transparent"
                           AbsoluteLayout.LayoutFlags="PositionProportional"
                           AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1">
       </ActivityIndicator>
```

```
<!--Indicator Label-->
<Label      x:Name="loading"
            Text="Loading Fires and Location..."
            TextColor="Black" FontSize="20"
            IsVisible="false"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.6,-1,-1">
</Label>

<!--Report Button-->
<Button     AbsoluteLayout.LayoutBounds="0.98,0.73,AutoSize,AutoSize"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            HeightRequest="52"
            WidthRequest="52"
            CornerRadius="25"
            BackgroundColor="LightGray"
            BorderColor="Black"
            BorderWidth="3"
            ImageSource="flame.png"
            Clicked="ReportFire_Clicked"
            IsVisible="true"
            x:Name="Report_Clicked">
</Button>

<!--Search Button-->
<Button     AbsoluteLayout.LayoutBounds="0.98, 0.84, AutoSize, AutoSize"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            HeightRequest="52"
            WidthRequest="52"
            CornerRadius="25"
            BackgroundColor="LightGray"
            BorderColor="Black"
            BorderWidth="3"
            ImageSource="search.png"
            x:Name="searchPopup"
            Clicked="Search_Clicked"
            IsVisible="true">
</Button>

<!--Search Popup-->
<Frame      x:Name="popupSearch"
            BackgroundColor="Transparent"
            Padding="10, 10"
            IsVisible="false"
            AbsoluteLayout.LayoutBounds="0, 0, 1, 1"
            AbsoluteLayout.LayoutFlags="All"
            BorderColor="Black"
            HasShadow="True">

    <Frame      VerticalOptions="Center"
                HorizontalOptions="Center"
```

```
        BorderColor="Black"
        WidthRequest="300"
        HeightRequest="500"
        CornerRadius="15"
        HasShadow="True">

<Grid   VerticalOptions="FillAndExpand"
        BackgroundColor="White"
        RowSpacing="7"
        ColumnSpacing="0">

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

    <Image      Grid.Row="0"
                HorizontalOptions="End"
                HeightRequest="20"
                WidthRequest="20"
                Source="cancel.png"
                Aspect="AspectFill">
        <Image.GestureRecognizers>
            <TapGestureRecognizer   Tapped="RemovePopupTapped"
                                    NumberOfTapsRequired="1">
            </TapGestureRecognizer>
        </Image.GestureRecognizers>
    </Image>

    <Entry      Placeholder="Search a Location"

                FontSize="18"
                x:Name="originEntry"
                ClassId="origin"
                ReturnType="Search"
                HorizontalOptions="Start"
                TextColor="Black"
                Grid.Row="1"
                WidthRequest="205"
                PlaceholderColor="White"
                BackgroundColor="LightGray"
                ClearButtonVisibility="WhileEditing"
                IsEnabled="True">
    </Entry>

    <Button     ImageSource="search.png"
                Clicked="SearchPlace_Clicked"
                BackgroundColor="LightGray"
                BorderColor="Black"
                BorderWidth="2"
                CornerRadius="10"
                HorizontalOptions="End"
                Grid.Row="1"
                WidthRequest="80">
         <Button.Background>
             <LinearGradientBrush EndPoint="1,0">
                 <GradientStop Color="LightGray"  Offset="0.1" />
                 <GradientStop Color="white" Offset="1.0" />
             </LinearGradientBrush>
         </Button.Background>
     </Button>

     <ListView   VerticalOptions="FillAndExpand"
                BackgroundColor="Transparent"
                Grid.Row="2"
                Footer=""
                SelectedItem="{Binding PlaceSelected, Mode=TwoWay}"
                x:Name="list"
                ItemsSource="{Binding Places}"
                HasUnevenRows="true"
                SeparatorColor="Gray">

         <ListView.Triggers>
             <DataTrigger    TargetType="ListView"
                            Binding="{Binding ShowRecentPlaces}"
                            Value="True">

                 <Setter     Property="ItemsSource"
                            Value="{Binding RecentPlaces}" />
             </DataTrigger>
             <DataTrigger    TargetType="ListView"
                            Binding="{Binding ShowRecentPlaces}"
                            Value="False">

                 <Setter     Property="ItemsSource"
                            Value="{Binding Places}" />
             </DataTrigger>
         </ListView.Triggers>
         <ListView.Header>

             <StackLayout    x:Name="recentSearchText"
                            IsVisible="{Binding ShowRecentPlaces}">

                 <Label      LineBreakMode="WordWrap"
                            FontAttributes="Bold"
                            Margin="20,10"
                            x:Name="recentSearch"
                            Text="History">
                 </Label>
```

```
                            </StackLayout>
                        </ListView.Header>
                        <ListView.ItemTemplate>
                            <DataTemplate>
                                <ViewCell>
                                    <Grid   Padding="15"
                                            RowSpacing="2"
                                            ColumnSpacing="15">

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

                                        <Image      Source="pin.png"
                                                    HeightRequest="20"
                                                    WidthRequest="20"
                                                    VerticalOptions="Start"
                                                    Grid.Row="0"
                                                    Grid.Column="0"
                                                    Grid.RowSpan="2">
                                        </Image>

                                        <Label      LineBreakMode="MiddleTruncation"
                                                    Text="{Binding 
StructuredFormatting.MainText}"

                                                    Grid.Row="0"
                                                    Grid.Column="1"
                                                    TextColor="Black">
                                        </Label>

                                        <Label      LineBreakMode="MiddleTruncation"
                                                    Text="{Binding 
StructuredFormatting.SecondaryText}"

                                                    TextColor="Black"
                                                    Grid.Row="1"
                                                    Grid.Column="1">
                                        </Label>
                                    </Grid>
                                </ViewCell>
                            </DataTemplate>
                        </ListView.ItemTemplate>
                    </ListView>
                </Grid>
            </Frame>
        </Frame>

        <!--Location Button-->
        <Button     AbsoluteLayout.LayoutBounds="0.98,0.95,AutoSize,AutoSize"
                    AbsoluteLayout.LayoutFlags="PositionProportional"
                    HeightRequest="52"
                    WidthRequest="52"
                    CornerRadius="25"
                    BackgroundColor="LightGray"
                    BorderColor="Black"
                    BorderWidth="3"
                    ImageSource="pin.png"
                    Clicked="Location_Button_Clicked"
                    IsVisible="true"
                    x:Name="Location_Clicked"/>
    </AbsoluteLayout>
</ContentPage>
```

Figure 51 - MapView.xaml

3.1.5.16. MapView.xaml.cs

```
/* Author:      Jack McNally
 * Page Name:   MapView
 * Purpose:     Backend for Map View.
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms.GoogleMaps;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using Wildfire.Views;
using Xamarin.Essentials;
using Wildfire.Helper;
using Xamarin.Forms.Internals;
using Wildfire.Services;
using Wildfire.Models;
using System.Windows.Input;

namespace Wildfire.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MapView : ContentPage
    {
        public static int loginCount = 0;
        public static int locationCount = 0;
        public static int notificationCount = 0;
        public static int reportedIndicator = 0;
        public static Pin recent;
        public static int fireNotCount = 0;
        public static readonly BindableProperty FocusOriginCommandProperty =
           BindableProperty.Create(nameof(FocusOriginCommand), typeof(ICommand),
typeof(SearchView), null, BindingMode.TwoWay);

        public ICommand FocusOriginCommand
        {
            get { return (ICommand)GetValue(FocusOriginCommandProperty); }
            set { SetValue(FocusOriginCommandProperty, value); }
        }

        FirebaseHelper firebaseHelper = new FirebaseHelper();
        public MapView()
        {
            InitializeComponent();
        }

        protected override bool OnBackButtonPressed()
        {
            return true;
        }

        /// <summary>
        /// Loading Logic
        /// </summary>
        protected async override void OnAppearing()
        {
            base.OnAppearing();

            await ChecksFires();

            overlay.IsVisible = true;
            loading.IsVisible = true;
            map.IsVisible = false;
            Report_Clicked.IsVisible = false;
            searchPopup.IsVisible = false;
            Location_Clicked.IsVisible = false;

            if (locationCount == 0)
            {
                await LoadCurrentPosition();
                await LoadFires();
                overlay.IsVisible = false;
                loading.IsVisible = false;
                map.IsVisible = true;
                Report_Clicked.IsVisible = true;
                searchPopup.IsVisible = true;
                Location_Clicked.IsVisible = true;
                locationCount++;
            }
            else if (loginCount == 0)
            {
                await LoadCurrentPosition();
                await LoadFires();

                overlay.IsVisible = false;
                loading.IsVisible = false;
                map.IsVisible = true;
                Report_Clicked.IsVisible = true;
                searchPopup.IsVisible = true;
                Location_Clicked.IsVisible = true;
                loginCount++;
            }
            else
            {
                await LoadFires();
                overlay.IsVisible = false;
                loading.IsVisible = false;
                map.IsVisible = true;
                Report_Clicked.IsVisible = true;
                searchPopup.IsVisible = true;
                Location_Clicked.IsVisible = true;
            }
        }

        protected override void OnBindingContextChanged()
        {
            base.OnBindingContextChanged();
            if (BindingContext != null)
            {
                FocusOriginCommand = new Command(OnOriginFocus);
            }
        }

        void OnOriginFocus()
        {
            originEntry.Focus();
        }

        async Task ChecksFires()
        {
            if(reportedIndicator == 0)
            {

            }
            if(reportedIndicator == 2)
            {
                reportedIndicator = 0;
                map.Pins.Remove(recent);
                OnPropertyChanged();
            }
            if(reportedIndicator == 1)
            {
                reportedIndicator = 0;
            }
        }
        /// <summary>
        /// Load Fires
        /// </summary>

        async Task LoadFires()
        {
            try
            {
                // Get Fires from Firebase
                var displayFires = await firebaseHelper.GetAllFires();
                foreach (var i in displayFires)
                {
                    if (i.Description.Length <= 10)
                    {
                        Pin newFire = new Pin()
                        {
                            Icon = (Device.RuntimePlatform == Device.Android) ?
BitmapDescriptorFactory.FromBundle("FlamePins.png") :
BitmapDescriptorFactory.FromView(new Image() { Source = "FlamePins.png",
WidthRequest = 20, HeightRequest = 20 }),
                            Label = i.Description.ToString(),
                            Position = new Position(Convert.ToDouble(i.Latitude),
Convert.ToDouble(i.Longitude)),
                            Address = i.PlaceName.ToString(),
                            Tag = i.FireID.ToString()
                        };
                        map.PinClicked += (sender, e) =>
                        {

                        };
                        map.Pins.Add(newFire);
                    }
                    else if (i.Description.Length > 10)
                    {
                        string newW = String.Concat(i.Description.Select((c, j) => j
> 0 && (j % 20) == 0 ? c.ToString() + Environment.NewLine : c.ToString()));

                        Pin newFire = new Pin()
                        {
                            Icon = (Device.RuntimePlatform == Device.Android) ?
BitmapDescriptorFactory.FromBundle("FlamePins.png") :
BitmapDescriptorFactory.FromView(new Image() { Source = "FlamePins.png",
WidthRequest = 20, HeightRequest = 20 }),
                            Label = newW.ToString(),
                            Position = new Position(Convert.ToDouble(i.Latitude),
Convert.ToDouble(i.Longitude)),
                            Address = i.PlaceName.ToString(),
                            Tag = i.FireID.ToString()
                        };
                        map.PinClicked += (sender, e) =>
                        {

                        };
                        map.Pins.Add(newFire);
                    }
                }
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        /// <summary>
        /// Load current position
        /// </summary>
        async Task LoadCurrentPosition()
        {
            try
            {
                // Permissions
                var phonePermissions = await
Permissions.CheckStatusAsync<Permissions.Phone>();
                phonePermissions = await
Permissions.CheckStatusAsync<Permissions.LocationWhenInUse>();

                if (phonePermissions != PermissionStatus.Granted)
                {
                    phonePermissions = await
Permissions.RequestAsync<Permissions.Phone>();
                    phonePermissions  = await
Permissions.RequestAsync<Permissions.LocationWhenInUse>();
                }

                if (phonePermissions != PermissionStatus.Granted)
                {
                    return;
                }

                var permissions = await
Permissions.CheckStatusAsync<Permissions.LocationWhenInUse>();

                if (permissions != PermissionStatus.Granted)
                {
                    permissions = await
Permissions.RequestAsync<Permissions.LocationWhenInUse>();
                }

                if (permissions != PermissionStatus.Granted)
                {
                    return;
                }

                var permission = await
Permissions.CheckStatusAsync<Permissions.Camera>();

                if (permission != PermissionStatus.Granted)
                {
                    permission = await
Permissions.RequestAsync<Permissions.Camera>();
                }

                if (permission != PermissionStatus.Granted)
                {
                    return;
                }

                var storagePermissionRead = await
Permissions.CheckStatusAsync<Permissions.StorageRead>();

                if (storagePermissionRead != PermissionStatus.Granted)
                {
                    storagePermissionRead = await
Permissions.RequestAsync<Permissions.StorageRead>();
                }

                if (storagePermissionRead != PermissionStatus.Granted)
                {
                    return;
                }

                var storagePermissionWrite = await
Permissions.CheckStatusAsync<Permissions.StorageWrite>();

                if (storagePermissionWrite != PermissionStatus.Granted)
                {
                    storagePermissionWrite = await
Permissions.RequestAsync<Permissions.StorageWrite>();
                }

                if (storagePermissionWrite != PermissionStatus.Granted)
                {
                    return;
                }

                // Get Location
                var location = await Geolocation.GetLocationAsync();
                Circle circle = new Circle()
                {
                    Center = new Position(location.Latitude, location.Longitude),
                    Radius = new Distance(Convert.ToDouble(SettingsView.radius) *
1000),
                    StrokeColor = Color.FromHex("#88FF0000"),
                    StrokeWidth = 4,
                    FillColor = Color.FromHex("#88FFC0CB"),
                    IsClickable = true
                };
                map.Circles.Clear();
                map.Pins.Clear();

                if (location != null)
                {
                    //Notification SET + Raduis SET
                    if (SettingsView.isChecked == true && SettingsView.radius !=
null)
                    {
                        Pin newLoc = new Pin()
                        {
                            Label = "Current Location",
                            Position = new Position(location.Latitude,
location.Longitude)
                        };
                        map.Pins.Add(newLoc);
                        map.Circles.Add(circle);
                        map.MoveToRegion(MapSpan.FromCenterAndRadius(new
Position(location.Latitude, location.Longitude), Distance.FromMeters(2000)));
                        var allFires = await firebaseHelper.GetAllFires();
                        foreach (var i in allFires)
                        {
                            var Loc = new Location(Convert.ToDouble(i.Latitude),
Convert.ToDouble(i.Longitude));

                            var comp = Location.CalculateDistance(location.Latitude,
location.Longitude, Loc, DistanceUnits.Kilometers);
                            comp.ToString();
                            // Count fires within the radius
                            try
                            {
                                if (Convert.ToDouble(comp) <=
Convert.ToDouble(SettingsView.radius))
                                {
                                    fireNotCount++;
                                }
                                else
                                {
                                }
                            }
                            catch (Exception ex)
                            {
                                await DisplayAlert("Faild", ex.Message, "OK");
                            }
                        }
                        // Send Notification
                        try
                        {
                              if (LoginPageView.token != null)
                              {
                                  if (notificationCount == 0)
                                  {
                                      if (fireNotCount == 1)
                                      {

DependencyService.Get<INotification>().CreateNotification("Wildfire", "A fire is
active in your area.");
                                          notificationCount++;
                                      }
                                      else
                                      {

DependencyService.Get<INotification>().CreateNotification("Wildfire",
fireNotCount.ToString() + " fires are active in your area.");
                                          notificationCount++;
                                      }
                                  }
                                  else
                                  {
                                  }
                              }
                              else
                              {
                                  if (notificationCount == 0)
                                  {
                                      if (fireNotCount == 1)
                                      {

DependencyService.Get<INotification>().CreateNotification("Wildfire", "A fire is
active in your area.");
                                          notificationCount++;
                                      }
                                      else if (fireNotCount == 0)
                                      {
                                      }
                                      else
                                      {

DependencyService.Get<INotification>().CreateNotification("Wildfire",
fireNotCount.ToString() + " fires are active in your area.");
                                          notificationCount++;
                                      }
                                  }
                                  else
                                  {
                                  }

                              }
                          }
                          catch (Exception ex)
```

```
                        {
                            ex.Message.ToString();
                        }

                    }
                    //Notification SET + Raduis NOT SET
                    else if (SettingsView.isChecked == true && SettingsView.radius
== null)
                    {

                        Pin newLoc = new Pin()
                        {
                            Label = "Current Location",
                            Position = new Position(location.Latitude,
location.Longitude)
                        };
                        map.Pins.Add(newLoc);
                        map.MoveToRegion(MapSpan.FromCenterAndRadius(new
Position(location.Latitude, location.Longitude), Distance.FromMeters(2000)));
                        var allFires = await firebaseHelper.GetAllFires();
                        foreach (var i in allFires)
                        {
                            var Loc = new Location(Convert.ToDouble(i.Latitude),
Convert.ToDouble(i.Longitude));

                            var comp = Location.CalculateDistance(location.Latitude,
location.Longitude, Loc, DistanceUnits.Kilometers);
                            comp.ToString();
                            try
                            {
                                if (Convert.ToDouble(comp) <= Convert.ToDouble(5))
                                {
                                    fireNotCount++;
                                }
                                else
                                {
                                }
                            }
                            catch (Exception ex)
                            {
                                ex.Message.ToString();
                            }
                        }
                        // Send Notification
                        try
                        {
                            if (LoginPageView.token != null)
                            {
                                if (notificationCount == 0)
                                {
                                    if (fireNotCount == 1)
                                    {
```

```
DependencyService.Get<INotification>().CreateNotification("Wildfire", "A fire is 
active in your area.");
                                        notificationCount++;
                                    }
                                    else if (fireNotCount == 0)
                                    {

                                    }
                                    else
                                    {

DependencyService.Get<INotification>().CreateNotification("Wildfire", 
fireNotCount.ToString() + " fires are active in your area.");
                                        notificationCount++;
                                    }
                                }
                                else
                                {
                                }
                            }
                            else
                            {
                                if (notificationCount == 0)
                                {
                                    if (fireNotCount == 1)
                                    {

DependencyService.Get<INotification>().CreateNotification("Wildfire", "A fire is 
active in your area.");
                                        notificationCount++;
                                    }
                                    else
                                    {

DependencyService.Get<INotification>().CreateNotification("Wildfire", 
fireNotCount.ToString() + " fires are active in your area.");
                                        notificationCount++;
                                    }
                                }
                                else
                                {
                                }

                            }
                        }
                        catch (Exception ex)
                        {
                            ex.Message.ToString();
                        }

                    }
                    //Notification NOT SET  + Raduis SET
                    else if (SettingsView.isChecked == false && SettingsView.radius
!= null)
                    {

                        Pin newLoc = new Pin()
                        {
                            Label = "Current Location",
                            Position = new Position(location.Latitude,
location.Longitude)
                        };
                        map.Pins.Add(newLoc);
                        map.Circles.Add(circle);
                        map.MoveToRegion(MapSpan.FromCenterAndRadius(new
Position(location.Latitude, location.Longitude), Distance.FromMeters(2000)));
                        var allFires = await firebaseHelper.GetAllFires();
                        foreach (var i in allFires)
                        {
                            var Loc = new Location(Convert.ToDouble(i.Latitude),
Convert.ToDouble(i.Longitude));

                            var comp = Location.CalculateDistance(location.Latitude,
location.Longitude, Loc, DistanceUnits.Kilometers);
                            comp.ToString();
                            try
                            {

                                if (Convert.ToDouble(comp) <=
Convert.ToDouble(SettingsView.radius))
                                {
                                    fireNotCount++;
                                }
                                else
                                {

                                }
                            }
                            catch (Exception ex)
                            {
                                ex.Message.ToString();
                            }
                        }
                        //Send Notifications
                    }

                    //Notification NOT SET + Raduis NOT SET
                    else if (SettingsView.isChecked == false && SettingsView.radius
== null)
                    {

                        Pin newLoc = new Pin()
                        {
                            Label = "Current Location",
                            Position = new Position(location.Latitude,
location.Longitude)
                        };
                        map.Pins.Add(newLoc);
                        map.Circles.Remove(circle);
                        map.MoveToRegion(MapSpan.FromCenterAndRadius(new
Position(location.Latitude, location.Longitude), Distance.FromMeters(2000)));
                        var allFires = await firebaseHelper.GetAllFires();
                        foreach (var i in allFires)
                        {
                            var Loc = new Location(Convert.ToDouble(i.Latitude),
Convert.ToDouble(i.Longitude));

                            var comp = Location.CalculateDistance(location.Latitude,
location.Longitude, Loc, DistanceUnits.Kilometers);
                            comp.ToString();
                            try
                            {
                                if (Convert.ToDouble(comp) <= Convert.ToDouble(5))
                                {

                                }
                                else
                                {

                                }
                            }
                            catch (Exception ex)
                            {
                                ex.Message.ToString();
                            }

                        }
                        //Send Notification
                    }
                }
            }
            catch(Exception ex)
            {
                ex.Message.ToString();



                var permissionLocation = await
Permissions.CheckStatusAsync<Permissions.LocationWhenInUse>();
                var current = Connectivity.NetworkAccess;
                if (permissionLocation != PermissionStatus.Granted)
                {
                    await DisplayAlert("Error", "Please enable Location services on
your mobile device and restart the application", "ok");
                    throw new Exception();
                }
                if(current != NetworkAccess.Internet)
                {
                    await DisplayAlert("Error", "Please enable WiFi services on your
mobile device and restart the application", "ok");
                    throw new Exception();
                }
            }
        }

        // search button event handler
        private void Search_Clicked(object sender, EventArgs e)
        {
            try
            {
                Report_Clicked.IsVisible = false;
                searchPopup.IsVisible = false;
                Location_Clicked.IsVisible = false;
                popupSearch.IsVisible = true;
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Remove popup event handler
        void RemovePopupTapped(object sender, EventArgs e)
        {
            try
            {
                Report_Clicked.IsVisible = true;
                searchPopup.IsVisible = true;
                Location_Clicked.IsVisible = true;
                popupSearch.IsVisible = false;
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Map click event handler
        private async void map_MapClicked(object sender, MapClickedEventArgs e)
        {
            try
            {
                // Permission check
                var permissions = await
Permissions.CheckStatusAsync<Permissions.Phone>();

                if (permissions != PermissionStatus.Granted)
                {
                    permissions = await
Permissions.RequestAsync<Permissions.Phone>();
                }

                if (permissions != PermissionStatus.Granted)
                {
                    return;
                }
                var location = await Geolocation.GetLocationAsync();
                var plLat = location.Latitude;
                var plLong = location.Longitude;
                var placemark1 = await Geocoding.GetPlacemarksAsync(plLat, plLong);
                var placemarkDetails1 = placemark1?.FirstOrDefault();
                string locality1 = placemarkDetails1.AdminArea;
                string areaCode1 = placemarkDetails1.CountryCode;
                string Place1 = locality1 + ", " + areaCode1;

                Pin newFire = new Pin()
                {
                    Icon = (Device.RuntimePlatform == Device.Android) ?
BitmapDescriptorFactory.FromBundle("FlamePins.png") :
BitmapDescriptorFactory.FromView(new Image() { Source = "FlamePins.png",
WidthRequest = 20, HeightRequest = 20 }),
                    Label = "New Fire",
                    Position = new Position(e.Point.Latitude, e.Point.Longitude),
                    Address = Place1,
                    IsDraggable = true
                };

                map.Pins.Add(newFire);

                await Task.Delay(500);
                recent = newFire;
                var Lat = e.Point.Latitude;
                var Long = e.Point.Longitude;
                var Place = Place1;
                Lat.ToString();
                Long.ToString();
                await Navigation.PushModalAsync(new ReportFireInfoView(Lat, Long,
Place) { BindingContext = this.BindingContext }, false);
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Location Button Handler
        private async void Location_Button_Clicked(object sender, EventArgs e)
        {
            try
            {
                var location = await Geolocation.GetLocationAsync();
                if (location != null)
                {
                    Pin newLoc = new Pin()
                    {
                        Label = "Current Location",
                        Position = new Position(location.Latitude,
location.Longitude)
                    };
                    map.Pins.Add(newLoc);
                    map.MoveToRegion(MapSpan.FromCenterAndRadius(new
Position(location.Latitude, location.Longitude), Distance.FromMeters(2000)));
                }
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Report fire button Handler
        private async void ReportFire_Clicked(object sender, EventArgs e)
        {
            try
            {

                var permissions = await
Permissions.CheckStatusAsync<Permissions.Phone>();

                if (permissions != PermissionStatus.Granted)
                {
                    permissions = await
Permissions.RequestAsync<Permissions.Phone>();
                }

                if (permissions != PermissionStatus.Granted)
                {
                    return;
                }
                var location = await Geolocation.GetLocationAsync();
                var plLat = location.Latitude;
                var plLong = location.Longitude;
                var placemark1 = await Geocoding.GetPlacemarksAsync(plLat, plLong);
                var placemarkDetails1 = placemark1?.FirstOrDefault();
                string locality1 = placemarkDetails1.AdminArea;
                string areaCode1 = placemarkDetails1.CountryCode;
                string Place1 = locality1 + ", " + areaCode1;
                if (location != null)
                {
                    Pin newLoc = new Pin()
                    {
                        Icon = (Device.RuntimePlatform == Device.Android) ?
BitmapDescriptorFactory.FromBundle("FlamePins.png") :
BitmapDescriptorFactory.FromView(new Image() { Source = "FlamePins.png",
WidthRequest = 20, HeightRequest = 20 }),
                        Label = "New Fire",
                        Position = new Position(location.Latitude,
location.Longitude),
                        Address = Place1
                    };

                    map.Pins.Add(newLoc);
                    var Lat = location.Latitude;
                    var Long = location.Longitude;
                    reportedIndicator = 0;
                    recent = newLoc;
                    var placemarks = await Geocoding.GetPlacemarksAsync(Lat, Long);
                    var placemarkDetails = placemarks?.FirstOrDefault();
                    string areaCode = placemarkDetails.AdminArea;
                    string localityName = placemarkDetails.Locality;
                    string Place = localityName + " " + areaCode;
                    Lat.ToString();
                    Long.ToString();

                    await Navigation.PushModalAsync(new ReportFireInfoView(Lat,
Long, Place) { BindingContext = this.BindingContext }, false);
                }
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Map Clicked event handler
        private async void map_PinClicked(object sender, PinClickedEventArgs e)
        {
            try
            {
                if (LoginPageView.token == null)
                {

                }
                else
                {
                    await Task.Delay(100);
                    await Navigation.PushModalAsync(new
ResolveFireInfoView(e.Pin.Label, e.Pin.Address, e.Pin.Tag.ToString()), false);
                }
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }

        }

        // search place event handler
        public async void SearchPlace_Clicked(object sender, EventArgs e)
        {
            try
            {
                var search = originEntry.Text;
                var searchLocation = await Geocoding.GetLocationsAsync(search);

                var sourceLocations = searchLocation?.FirstOrDefault();
                if (sourceLocations != null)
                {
                    Location sourceCoordinates = new
Location(sourceLocations.Latitude, sourceLocations.Longitude);

                    Pin pin = new Pin()
                    {
                        Icon = (Device.RuntimePlatform == Device.Android) ?
BitmapDescriptorFactory.FromBundle("SearchPins.png") :
BitmapDescriptorFactory.FromView(new Image() { Source = "SearchPins.png",
WidthRequest = 20, HeightRequest = 20 }),
                        Type = PinType.Place,
                        Label = originEntry.Text,
                        Position = new Position(sourceCoordinates.Latitude,
sourceCoordinates.Longitude)
                    };
                    map.Pins.Add(pin);

                    map.MoveToRegion(MapSpan.FromCenterAndRadius(new
Position(sourceCoordinates.Latitude, sourceCoordinates.Longitude),
Distance.FromMeters(500)));
                    originEntry.Text = string.Empty;
                    search = string.Empty;
                    popupSearch.IsVisible = false;
                    Report_Clicked.IsVisible = true;
                    searchPopup.IsVisible = true;
                    Location_Clicked.IsVisible = true;
                }
                else
                {
                    search = string.Empty;
                    await DisplayAlert("Cant Find Location", "", "Yes");
                }
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }
```

```
    }
}
```

Figure 52 - MapView.xaml.cs

3.1.5.17. ProCurrentFireView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     ProCurrentFireaView
    Purpose:       Current fires View interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.ProCurrentFiresView">

   <!--Background-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="#F7D9BB"  Offset="0.1" />
           <GradientStop Color="White" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>
       <StackLayout>
           <StackLayout>
               <!--Header-->
               <Frame      BorderColor="LightGray"
                           Padding="7"
                           WidthRequest="400"
                           HeightRequest="80"
                           AbsoluteLayout.LayoutFlags="PositionProportional"
                           AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                           BackgroundColor="LightGray"
                           HasShadow="True"
                           CornerRadius="15"
                           IsVisible="true"
                           HorizontalOptions="Center">

                   <Frame      BorderColor="LightGray"
                               Padding="7"
                               HeightRequest="80"
                               WidthRequest="400"
                               AbsoluteLayout.LayoutFlags="PositionProportional"
                               AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                               BackgroundColor="LightGray"
                               HasShadow="True"
                               CornerRadius="15"
```

```
                        IsVisible="true"
                        HorizontalOptions="Center">
                <Frame.Background>
                    <LinearGradientBrush EndPoint="0,1">
                        <GradientStop Color="White"  Offset="0.9" />
                        <GradientStop Color="LightBlue" Offset="1.0" />
                    </LinearGradientBrush>
                </Frame.Background>

                <Label      Text="Current Fires"
                            FontSize="26"
                            HorizontalOptions="Center"
                            VerticalOptions="Center"
                            TextColor="#282828"
                            FontAttributes="Bold">
                </Label>
            </Frame>
        </Frame>
</StackLayout>

<!--ListView-->
<ListView   x:Name="currentFires"
            HasUnevenRows="True"
            RowHeight="70"
            SeparatorColor="Black"
            ItemTapped="CurrentFires_ItemTapped">
    <ListView.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#F7D9BB"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </ListView.Background>
    <ListView.ItemTemplate>
        <DataTemplate>
            <TextCell   Grid.Row="1"
                        Text="{Binding PlaceName}"
                        Detail="{Binding Time}"
                        TextColor="Black"
                        DetailColor="Gray">
            </TextCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>

<!--Back Button-->
<Button     x:Name="BackButton"
            Text="Back"
            HeightRequest="56"
            BackgroundColor="LightGray"
            FontSize="20"
            FontAttributes="Bold"
            BorderWidth="3"
```

```
                       BorderColor="Black"
                       TextColor="#282828"
                       ImageSource="backarrow.png"
                       ContentLayout="Left"
                       Padding="50,0,75,0"
                       Clicked="BackButton_Clicked">
               <Button.Background>
                   <LinearGradientBrush EndPoint="0,1">
                       <GradientStop Color="LightGray"  Offset="0.1" />
                       <GradientStop Color="white" Offset="1.0" />
                   </LinearGradientBrush>
               </Button.Background>
           </Button>
       </StackLayout>
   </ContentPage.Content>
</ContentPage>
```

Figure 53 - ProCurrentFireView.xaml

3.1.5.18. ProCurrentFireView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   ProCurrentFiresView
* Purpose:     Backend for Current Fires View.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Helper;
using Wildfire.Models;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class ProCurrentFiresView : ContentPage
   {
       FirebaseHelper firebaseHelper = new FirebaseHelper();
       public ProCurrentFiresView()
       {
           InitializeComponent();
       }
       /// <summary>
       /// Loading Logic
       /// </summary>
       protected async override void OnAppearing()
```

```
        {
            base.OnAppearing();
            var fires = await firebaseHelper.GetAllFires();
            currentFires.ItemsSource = fires;
        }

        // List Item tap event handler
        private async void CurrentFires_ItemTapped(object sender, ItemTappedEventArgs
e)
        {
            try
            {
                await Task.Delay(10);
                var fireInfo = e.Item as Fire;
                await Navigation.PushModalAsync(new
ProCurrrentSelectedView(fireInfo.WindDirection, fireInfo.PlaceName, fireInfo.FireID,
fireInfo.Time, fireInfo.Description),false);
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Back Button Event Handler
        private async void BackButton_Clicked(object sender, EventArgs e)
        {
            try
            {
                await App.Current.MainPage.Navigation.PopModalAsync();
            }
            catch (Exception ex)
            {
                ex.Message.ToString();
            }
        }
    }
}
```

Figure 54 - ProCurrentFireView.xaml.cs

3.1.5.19. ProCurrentSelectedView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     ProCurrentSelectedView
    Purpose:       Current fire selected View interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
```

```
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         x:Class="Wildfire.Views.ProCurrrentSelectedView">

<!--Background-->
<ContentPage.Background>
    <LinearGradientBrush EndPoint="0,1">
        <GradientStop Color="White" Offset="0.8" />
        <GradientStop Color="#ff8c19"  Offset="1.0" />
    </LinearGradientBrush>
</ContentPage.Background>

<!--Content Page-->
<ContentPage.Content>
    <ScrollView>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height="15"></RowDefinition>
                <RowDefinition Height="80"></RowDefinition>
                <RowDefinition Height="5"></RowDefinition>
                <RowDefinition Height="25"></RowDefinition>
                <RowDefinition Height="25"></RowDefinition>
                <RowDefinition Height="5"></RowDefinition>
                <RowDefinition Height="25"></RowDefinition>
                <RowDefinition Height="25"></RowDefinition>
                <RowDefinition Height="5"></RowDefinition>
                <RowDefinition Height="25"></RowDefinition>
                <RowDefinition Height="25"></RowDefinition>
                <RowDefinition Height="5"></RowDefinition>
                <RowDefinition Height="25"></RowDefinition>
                <RowDefinition Height="50"></RowDefinition>
                <RowDefinition Height="125"></RowDefinition>
                <RowDefinition Height="56"></RowDefinition>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"></ColumnDefinition>
                <ColumnDefinition Width="300"></ColumnDefinition>
                <ColumnDefinition Width="*"></ColumnDefinition>
            </Grid.ColumnDefinitions>

            <!--Header-->
            <Frame      BorderColor="LightGray"
                        Padding="7"
                        HeightRequest="80"
                        WidthRequest="300"
                        Grid.Row="1"
                        Grid.Column="1"
                        BackgroundColor="LightGray"
                        HasShadow="True"
                        CornerRadius="15"
                        IsVisible="true"
                        HorizontalOptions="Center">
    <Frame      BorderColor="LightGray"
                Padding="7"
                HeightRequest="80"
                WidthRequest="310"
                Grid.Row="1"
                Grid.Column="1"
                BackgroundColor="LightGray"
                HasShadow="True"
                CornerRadius="15"
                IsVisible="true"
                HorizontalOptions="Center">
        <Frame.Background>
            <LinearGradientBrush EndPoint="0,1">
                <GradientStop Color="White"  Offset="0.9" />
                <GradientStop Color="LightBlue" Offset="1.0" />
            </LinearGradientBrush>
        </Frame.Background>

        <Label      Text="Fire Information"
                    FontSize="26"
                    HorizontalOptions="Center"
                    VerticalOptions="Center"
                    TextColor="#282828"
                    FontAttributes="Bold">
        </Label>
    </Frame>
</Frame>

<!--Placename Label-->
<Label      Text="Placename"
            Grid.Row="3"
            TextColor="Black"
            FontSize=" 20"
            Grid.Column="1">
</Label>

<Label      x:Name="placeName"
            Grid.Row="4"
            TextColor="Black"
            FontSize="18"
            Grid.Column="1"
            Padding="10,0,0,0"
            WidthRequest="200"
            BackgroundColor="#2daff0">
    <Label.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue"  Offset="0.5" />
            <GradientStop Color="#2daff0" Offset="1.0" />
        </LinearGradientBrush>
    </Label.Background>
</Label>
```

```
<!--Time Found Label-->
<Label      Text="Time Found"
            Grid.Row="6"
            TextColor="Black"
            FontSize=" 20"
            Grid.Column="1">
</Label>

<Label      x:Name="time"
            Grid.Row="7"
            TextColor="Black"
            FontSize=" 18"
            Grid.Column="1"
            Padding="10,0,0,0"
            WidthRequest="200"
            BackgroundColor="#2daff0">
    <Label.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue"  Offset="0.5" />
            <GradientStop Color="#2daff0" Offset="1.0" />
        </LinearGradientBrush>
    </Label.Background>
</Label>

<!-- Wind Direction Label-->
<Label      Text="Wind Direction"
            Grid.Row="9"
            TextColor="Black"
            FontSize="20"
            Grid.Column="1">
</Label>

<Label      x:Name="windDir"
            Grid.Row="10"
            TextColor="Black"
            FontSize="18"
            Grid.Column="1"
            Padding="10,0,0,0"
            WidthRequest="200"
            BackgroundColor="#2daff0">
    <Label.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue"  Offset="0.5" />
            <GradientStop Color="#2daff0" Offset="1.0" />
        </LinearGradientBrush>
    </Label.Background>
</Label>

<!--Description Label-->
<Label      Text="Description"
            Grid.Row="12"
            TextColor="Black"
            FontSize="20"
            Grid.Column="1">
</Label>

<Label      x:Name="description"
            Grid.Row="13"
            TextColor="Black"
            FontSize="18"
            Grid.Column="1"
            Padding="10,0,0,0"
            WidthRequest="200"
            LineBreakMode="WordWrap"
            VerticalOptions="CenterAndExpand"
            BackgroundColor="#2daff0">
    <Label.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue"  Offset="0.5" />
            <GradientStop Color="#2daff0" Offset="1.0" />
        </LinearGradientBrush>
    </Label.Background>
</Label>

<Label      x:Name="fireID"
            Grid.Row="0"
            TextColor="White"
            FontSize="0"
            Grid.Column="0">
</Label>

<!--Image Import-->
<Image      x:Name="currentFire"
            Grid.Row="14"
            Grid.Column="1"
            HeightRequest="125"
            Source="photo.png">
</Image>

<!--Activity Indicator-->
<ActivityIndicator x:Name="overlay"
                   IsRunning="True"
                   IsVisible="True"
                   Color="#282828"
                   BackgroundColor="Transparent"
                   Grid.Column="1"
                   Grid.Row="14">
</ActivityIndicator>

<!--Back Button-->
<Button     x:Name="BackButton"
            Text="Back"
            WidthRequest="100"
            Grid.Row="15"
```

```
                                Grid.Column="1"
                                BackgroundColor="LightGray"
                                FontSize="20"
                                FontAttributes="Bold"
                                CornerRadius="10"
                                BorderWidth="3"
                                BorderColor="Black"
                                TextColor="Black"
                                ImageSource="backarrow.png"
                                ContentLayout="Left"
                                Padding="40,0,75,0"
                                Clicked="BackButton_Clicked">
                        <Button.Background>
                            <LinearGradientBrush EndPoint="0,1">
                                <GradientStop Color="LightGray"  Offset="0.1" />
                                <GradientStop Color="white" Offset="1.0" />
                            </LinearGradientBrush>
                        </Button.Background>
                    </Button>
                </Grid>
            </ScrollView>
        </ContentPage.Content>
</ContentPage>
```

Figure 55 - ProCurrentSelectedView.xaml

3.1.5.20. ProCurrentSelectedView.xaml.cs

```
/* Author:      Jack McNally
 * Page Name:   ProCurrentSelectedView
 * Purpose:     Backend for Current Selected fires View.
 */
using Firebase.Storage;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Helper;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class ProCurrrentSelectedView : ContentPage
   {
       FirebaseHelper firebaseHelper = new FirebaseHelper();
       public ProCurrrentSelectedView(string WindDirection, string PlaceName, string 
FireID, string Time, string Description)
       {
           InitializeComponent();

           placeName.Text = PlaceName;
           description.Text = Description;
           time.Text = Time;
           windDir.Text = WindDirection;
           fireID.Text = FireID;
       }
       /// <summary>
       /// Loading Logic
       /// </summary>
       protected async override void OnAppearing()
       {
           base.OnAppearing();
           overlay.IsVisible = true;
           currentFire.IsVisible = false;
           await LoadImage();
           currentFire.IsVisible = true;
           overlay.IsVisible = false;
       }

       /// <summary>
       /// Loading images from firebase
       /// </summary>
       public async Task LoadImage()
       {
           try
           {
               var filename = fireID.Text;
               var webClient = new WebClient();
               var storageImage = await new 
FirebaseStorage("driven-bulwark-297919.appspot.com")
                   .Child("Fires")
                   .Child(filename + ".jpeg")
                   .GetDownloadUrlAsync();
               string imgurl = storageImage;
               byte[] imgbytes = webClient.DownloadData(imgurl);
               currentFire.Source = ImageSource.FromStream(() => new 
MemoryStream(imgbytes));
           }
           catch(Exception ex)
           {
               ex.Message.ToString();
           }
       }

       // Back Button Event Handler
       private async void BackButton_Clicked(object sender, EventArgs e)
       {
           try
           {
               await App.Current.MainPage.Navigation.PopModalAsync();
           }
           catch(Exception ex)
           {
               ex.Message.ToString();
           }
       }
   }
}
```

Figure 56 - ProCurrentSelectedView.xaml.cs

3.1.5.21. ProFireInfoView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     ProFireInfoView
    Purpose:       Info View Fires interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.ProFireInfoView">

   <!--Background-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="#ff8c19"  Offset="0.1" />
           <GradientStop Color="#F7D9BB" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>
       <ScrollView>
           <Grid>
               <Grid.RowDefinitions>
                   <RowDefinition Height="*"></RowDefinition>
                   <RowDefinition Height="130"></RowDefinition>
                   <RowDefinition Height="20"></RowDefinition>
                   <RowDefinition Height="130"></RowDefinition>
                   <RowDefinition Height="*"></RowDefinition>
                   <RowDefinition Height="56"></RowDefinition>
                   <RowDefinition Height="*"></RowDefinition>
               </Grid.RowDefinitions>
               <Grid.ColumnDefinitions>
                   <ColumnDefinition Width="*"/>
                   <ColumnDefinition Width="300"/>
                   <ColumnDefinition Width="*"/>
               </Grid.ColumnDefinitions>

               <!-- Current Fires Button-->
               <Button     Grid.Row="1"
                          Grid.Column="1"
                          BorderColor="Black"
                          BorderWidth="3"
                          FontSize="25"
                          BackgroundColor="LightBlue"
                          CornerRadius="25"
                          Text="Current Fires"
                          TextColor="Black"
                          FontAttributes="Italic"
                          x:Name="CurrentFires"
                          Clicked="CurrentFires_Clicked"
                          IsVisible="true"
                          Padding="0,0,20,0"
                          ContentLayout="Right"
                          ImageSource="currents.png" >
                   <Button.Background>
                       <LinearGradientBrush EndPoint="1,0">
                           <GradientStop Color="LightBlue" Offset="0.3" />
                           <GradientStop Color="LightSalmon" Offset="1.0" />
                       </LinearGradientBrush>
                   </Button.Background>
               </Button>

               <!--Resolved Fires Button-->
               <Button     Grid.Row="3"
                          Grid.Column="1"
                          BorderColor="Black"
                          BorderWidth="3"
                          FontSize="25"
                          BackgroundColor="LightBlue"
                          CornerRadius="25"
                          Text="Resolved Fires"
                          TextColor="Black"
                          FontAttributes="Italic"
                          x:Name="ResolvedFires"
                          Clicked="ResolvedFires_Clicked"
                          IsVisible="true"
                          Padding="0,0,20,0"
                          ContentLayout="Right"
                          ImageSource="resolve.png">
                   <Button.Background>
                       <LinearGradientBrush EndPoint="1,0">
                           <GradientStop Color="LightBlue" Offset="0.3" />
                           <GradientStop Color="LightSalmon" Offset="1.0" />
                       </LinearGradientBrush>
                   </Button.Background>
               </Button>
```

```
                <!--Back Button-->
                <Button     Grid.Row="5"
                            Grid.Column="1"
                            BorderColor="Black"
                            BorderWidth="3"
                            FontSize="20"
                            BackgroundColor="LightGray"
                            CornerRadius="10"
                            Text="Back"
                            TextColor="Black"
                            FontAttributes="Bold"
                            x:Name="BackButton"
                            ImageSource="backarrow.png"
                            ContentLayout="Left"
                            Padding="50,0,75,0"
                            Clicked="BackButton_Clicked">
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0,1">
                            <GradientStop Color="LightGray"  Offset="0.1" />
                            <GradientStop Color="white" Offset="1.0" />
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
            </Grid>
           </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Figure 57 - ProFireInfoView.xaml

3.1.5.22. ProFireInfoView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   ProFireInfoView
* Purpose:     Backend for Info View fires.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
```

```
    public partial class ProFireInfoView : ContentPage
    {
        public ProFireInfoView()
        {
            InitializeComponent();
        }

        // Current Fires Button Event Handler
        private async void CurrentFires_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PushModalAsync(new ProCurrentFiresView());
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Back Button Event Handler
        private async void BackButton_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PopModalAsync();
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Resolved Button Event Handler
        private async void ResolvedFires_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PushModalAsync(new ProResFireView());
            }
            catch (Exception ex)
            {
                ex.Message.ToString();
            }
        }
    }
}
```

Figure 58 - ProFireInfoView.xaml.cs

3.1.5.23. ProResFireView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     ProResFireView
    Purpose:       Resolved fires View interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.ProResFireView">

   <!--Background-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="#F7D9BB"  Offset="0.1" />
           <GradientStop Color="White" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>
       <StackLayout>
           <StackLayout>

               <!--Header-->
               <Frame      BorderColor="LightGray"
                           Padding="7"
                           WidthRequest="400"
                           HeightRequest="80"
                           AbsoluteLayout.LayoutFlags="PositionProportional"
                           AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                           BackgroundColor="LightGray"
                           HasShadow="True"
                           CornerRadius="15"
                           IsVisible="true"
                           HorizontalOptions="Center">

                   <Frame      BorderColor="LightGray"
                               Padding="7"
                               HeightRequest="80"
                               WidthRequest="400"
                               AbsoluteLayout.LayoutFlags="PositionProportional"
                               AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                               BackgroundColor="LightGray"
                               HasShadow="True"
                               CornerRadius="15"
                               IsVisible="true"
                               HorizontalOptions="Center">
                       <Frame.Background>
                           <LinearGradientBrush EndPoint="0,1">
                               <GradientStop Color="White"  Offset="0.9" />
```

```
                    <GradientStop Color="LightBlue" Offset="1.0" />
                </LinearGradientBrush>
            </Frame.Background>

            <Label      Text="Resolved Fires"
                        FontSize="26"
                        HorizontalOptions="Center"
                        VerticalOptions="Center"
                        TextColor="#282828"
                        FontAttributes="Bold">
            </Label>
        </Frame>
    </Frame>
</StackLayout>

<!--ListView-->
<ListView   x:Name="resFires"
            HasUnevenRows="True"
            RowHeight="70"
            SeparatorColor="Black"
            ItemTapped="resFires_ItemTapped">
    <ListView.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#F7D9BB"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </ListView.Background>
    <ListView.ItemTemplate>
        <DataTemplate>
            <TextCell   Grid.Row="1"
                        Text="{Binding PlaceName}"
                        Detail="{Binding Time}"
                        TextColor="Black"
                        DetailColor="Gray">
            </TextCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>

<!--Back Button-->
<Button     x:Name="BackButton"
            Text="Back"
            WidthRequest="100"
            HeightRequest="50"
            BackgroundColor="LightGray"
            FontSize="20"
            FontAttributes="Bold"
            BorderWidth="3"
            BorderColor="Black"
            TextColor="Black"
            ImageSource="backarrow.png"
            ContentLayout="Left"
```

```
                        Padding="50,0,75,0"
                        Clicked="BackButton_Clicked">
                <Button.Background>
                    <LinearGradientBrush EndPoint="0,1">
                        <GradientStop Color="LightGray"  Offset="0.1" />
                        <GradientStop Color="white" Offset="1.0" />
                    </LinearGradientBrush>
                </Button.Background>
            </Button>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Figure 59 - ProResFireView.xaml

5.1.5.24. ProResFireView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   ProResfireView
* Purpose:     Backend for Res Fire View.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Helper;
using Wildfire.Models;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class ProResFireView : ContentPage
   {
       FirebaseHelper firebaseHelper = new FirebaseHelper();
       public ProResFireView()
       {
           InitializeComponent();
       }
       /// <summary>
       /// Loading Logic
       /// </summary>
       protected async override void OnAppearing()
       {
           base.OnAppearing();
           var fires = await firebaseHelper.GetResolvedFires();
           resFires.ItemsSource = fires;
```

```
        }

        // // List Item tap event handler
        private async void resFires_ItemTapped(object sender, ItemTappedEventArgs e)
        {
            try
            {
                await Task.Delay(50);
                var fireInfo = e.Item as Fire;
                await App.Current.MainPage.Navigation.PushModalAsync(new
ProResSelectedView(fireInfo.PlaceName, fireInfo.FireID, fireInfo.Time,
fireInfo.ResolvedDescription), false);
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Back Button Event Handler
        private async void BackButton_Clicked(object sender, EventArgs e)
        {
            try
            {
                await App.Current.MainPage.Navigation.PopModalAsync();
            }
            catch (Exception ex)
            {
                ex.Message.ToString();
            }
        }
    }
}
```

Figure 60 - ProResFireView.xaml.cs

5.1.5.25. ProResSelectedView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     ProResSelectedView
    Purpose:       Resolved fire selected View interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.ProResSelectedView">

   <!--Background-->
   <ContentPage.Background>
```

```
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="White" Offset="0.8" />
            <GradientStop Color="#ff8c19"  Offset="1.0" />
        </LinearGradientBrush>
    </ContentPage.Background>

    <!--Content Page-->
    <ContentPage.Content>
        <ScrollView>
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="15"></RowDefinition>
                    <RowDefinition Height="70"></RowDefinition>
                    <RowDefinition Height="5"></RowDefinition>
                    <RowDefinition Height="25"></RowDefinition>
                    <RowDefinition Height="25"></RowDefinition>
                    <RowDefinition Height="5"></RowDefinition>
                    <RowDefinition Height="25"></RowDefinition>
                    <RowDefinition Height="25"></RowDefinition>
                    <RowDefinition Height="5"></RowDefinition>
                    <RowDefinition Height="25"></RowDefinition>
                    <RowDefinition Height="25"></RowDefinition>
                    <RowDefinition Height="5"></RowDefinition>
                    <RowDefinition Height="5"></RowDefinition>
                    <RowDefinition Height="1"></RowDefinition>
                    <RowDefinition Height="120"></RowDefinition>
                    <RowDefinition Height="20"></RowDefinition>
                    <RowDefinition Height="56"></RowDefinition>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"></ColumnDefinition>
                    <ColumnDefinition Width="300"></ColumnDefinition>
                    <ColumnDefinition Width="*"></ColumnDefinition>
                </Grid.ColumnDefinitions>

                <!--Header-->
                <Frame      BorderColor="LightGray"
                            Padding="7"
                            HeightRequest="80"
                            WidthRequest="300"
                            Grid.Row="1"
                            Grid.Column="1"
                            BackgroundColor="LightGray"
                            HasShadow="True"
                            CornerRadius="15"
                            IsVisible="true"
                            HorizontalOptions="Center">

                    <Frame      BorderColor="LightGray"
                                Padding="7"
                                HeightRequest="80"
                                WidthRequest="310"
                                Grid.Row="1"
                                Grid.Column="1"
                                BackgroundColor="LightGray"
                                HasShadow="True"
                                CornerRadius="15"
                                IsVisible="true"
                                HorizontalOptions="Center">
                        <Frame.Background>
                            <LinearGradientBrush EndPoint="0,1">
                                <GradientStop Color="White"  Offset="0.9" />
                                <GradientStop Color="LightBlue" Offset="1.0" />
                            </LinearGradientBrush>
                        </Frame.Background>

                        <Label      Text="Resolved Fire Information"
                                    FontSize="22"
                                    HorizontalOptions="Center"
                                    VerticalOptions="Center"
                                    TextColor="#282828"
                                    FontAttributes="Bold">
                        </Label>
                    </Frame>
                </Frame>

                <!--PLacename Label-->
                <Label      Text="Placename"
                            Grid.Row="3"
                            TextColor="Black"
                            FontSize=" 20"
                            Grid.Column="1">
                </Label>

                <Label      x:Name="placeName"
                            Grid.Row="4"
                            TextColor="Black"
                            FontSize="18"
                            Grid.Column="1"
                            Padding="10,0,0,0"
                            WidthRequest="200"
                            BackgroundColor="#2daff0">
                    <Label.Background>
                        <LinearGradientBrush EndPoint="1,0">
                            <GradientStop Color="LightBlue"  Offset="0.5" />
                            <GradientStop Color="#2daff0" Offset="1.0" />
                        </LinearGradientBrush>
                    </Label.Background>
                </Label>

                <!--Time Found Label-->
                <Label      Text="Time Found"
                            Grid.Row="6"
                            TextColor="Black"
                            FontSize=" 20"
                            Grid.Column="1">
                </Label>

                <Label      x:Name="time"
                            Grid.Row="7"
                            TextColor="Black"
                            FontSize=" 18"
                            Grid.Column="1"
                            Padding="10,0,0,0"
                            WidthRequest="200"
                            BackgroundColor="#2daff0">
                    <Label.Background>
                        <LinearGradientBrush EndPoint="1,0">
                            <GradientStop Color="LightBlue"  Offset="0.5" />
                            <GradientStop Color="#2daff0" Offset="1.0" />
                        </LinearGradientBrush>
                    </Label.Background>
                </Label>

                <!--Description Label-->
                <Label      Text="Description"
                            Grid.Row="9"
                            TextColor="Black"
                            FontSize="20"
                            Grid.Column="1">
                </Label>

                <Label      x:Name="description"
                            Grid.Row="10"
                            TextColor="Black"
                            FontSize="18"
                            Grid.Column="1"
                            Padding="10,0,0,0"
                            WidthRequest="200"
                            VerticalOptions="CenterAndExpand"
                            BackgroundColor="#2daff0">
                    <Label.Background>
                        <LinearGradientBrush EndPoint="1,0">
                            <GradientStop Color="LightBlue"  Offset="0.5" />
                            <GradientStop Color="#2daff0" Offset="1.0" />
                        </LinearGradientBrush>
                    </Label.Background>
                </Label>

                <Label      x:Name="fireID"
                            Grid.Row="0"
                            TextColor="White"
                            FontSize=" 0"
                            Grid.Column="0">
                </Label>
```

```
                <!--Image Import-->
                <Image      x:Name="currentFire"
                            Grid.Row="14"
                            Grid.Column="1"
                            HeightRequest="125"
                            Source="photo.png">
                </Image>

                <!--Activity Indicator-->
                <ActivityIndicator x:Name="overlay"
                                   IsRunning="True"
                                   IsVisible="True"
                                   Color="#282828"
                                   BackgroundColor="Transparent"
                                   Grid.Column="1"
                                   Grid.Row="14">
                </ActivityIndicator>

                <!--Back Button-->
                <Button     x:Name="BackButton"
                            Text="Back"
                            WidthRequest="100"
                            Grid.Row="16"
                            Grid.Column="1"
                            BackgroundColor="LightGray"
                            FontSize="20"
                            FontAttributes="Bold"
                            CornerRadius="10"
                            BorderWidth="3"
                            BorderColor="Black"
                            TextColor="Black"
                            ImageSource="backarrow.png"
                            ContentLayout="Left"
                            Padding="40,0,75,0"
                            Clicked="BackButton_Clicked">
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0,1">
                            <GradientStop Color="LightGray"  Offset="0.1" />
                            <GradientStop Color="white" Offset="1.0" />
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Figure 61 - ProResSelectedView.xaml.

5.1.5.26. ProResSelectedView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   ProResSelectedView
* Purpose:     Backend for Resolved fire selected view.
*/
using Firebase.Storage;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Helper;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class ProResSelectedView : ContentPage
   {
       FirebaseHelper firebaseHelper = new FirebaseHelper();
       public ProResSelectedView(string PlaceName, string FireID, string Time,
string ResDescription)
       {
           InitializeComponent();
           placeName.Text = PlaceName;
           time.Text = Time;
           fireID.Text = FireID;
           description.Text = ResDescription;
       }

       /// <summary>
       /// Loading Logic
       /// </summary>
       protected async override void OnAppearing()
       {
           base.OnAppearing();
           overlay.IsVisible = true;
           currentFire.IsVisible = false;
           await LoadImage();
           currentFire.IsVisible = true;
           overlay.IsVisible = false;
       }
       /// <summary>
       /// Load image task from firebase
       /// </summary>
       public async Task LoadImage()
       {
           try
           {
               var filename = fireID.Text;
               var webClient = new WebClient();
               var storageImage = await new
FirebaseStorage("driven-bulwark-297919.appspot.com")
                   .Child("Fires")
                   .Child(filename + "(new)" + ".jpeg")
                   .GetDownloadUrlAsync();
               string imgurl = storageImage;
               byte[] imgbytes = webClient.DownloadData(imgurl);
               currentFire.Source = ImageSource.FromStream(() => new
MemoryStream(imgbytes));
            }
            catch(Exception ex)
            {
               ex.Message.ToString();
            }
        }

       // Back Button Event Handler
       private async void BackButton_Clicked(object sender, EventArgs e)
       {
           try
           {
               await App.Current.MainPage.Navigation.PopModalAsync();
           }
           catch (Exception ex)
           {
               ex.Message.ToString();
           }
       }
   }
}
```

Figure 62 - ProResSelectedView.xaml.cs

5.1.5.27. ReportFireInfoView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
     Page Name:     ReportFireInfoView
     Purpose:       Report fire interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Wildfire.Views.ReportFireInfoView">

    <!--Background-->
    <ContentPage.Background>
        <LinearGradientBrush EndPoint="0,1">
```

```
            <GradientStop Color="White" Offset="0.1" />
            <GradientStop Color="#F7D9BB"  Offset="1.0" />
        </LinearGradientBrush>
    </ContentPage.Background>

    <!--Content Page-->
    <ContentPage.Content>
        <StackLayout>
            <ScrollView>
                <Grid RowSpacing="8">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="0"/>
                        <RowDefinition Height="84"/>
                        <RowDefinition Height="0"/>
                        <RowDefinition Height="32"/>
                        <RowDefinition Height="32" />
                        <RowDefinition Height="0"/>
                        <RowDefinition Height="25"/>
                        <RowDefinition Height="45"/>
                        <RowDefinition Height="0"/>
                        <RowDefinition Height="25"/>
                        <RowDefinition Height="70"/>
                        <RowDefinition Height="56"/>
                        <RowDefinition Height="56"/>
                        <RowDefinition Height="90"/>
                        <RowDefinition Height="5"/>
                        <RowDefinition Height="56"/>
                        <RowDefinition Height="56"/>
                        <RowDefinition Height="5"/>
                    </Grid.RowDefinitions>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*"/>
                        <ColumnDefinition Width="300"/>
                        <ColumnDefinition Width="*"/>
                    </Grid.ColumnDefinitions>

                    <!--Header-->
                    <Frame      BorderColor="LightGray"
                                Padding="7"
                                HeightRequest="80"
                                WidthRequest="300"
                                Grid.Row="1"
                                Grid.Column="1"
                                BackgroundColor="LightGray"
                                HasShadow="True"
                                CornerRadius="15"
                                IsVisible="true"
                                HorizontalOptions="Center">

                        <Frame      BorderColor="LightGray"
                                    Padding="7"
                                    HeightRequest="80"
                WidthRequest="310"
                Grid.Row="1"
                Grid.Column="1"
                BackgroundColor="LightGray"
                HasShadow="True"
                CornerRadius="15"
                IsVisible="true"
                HorizontalOptions="Center">
        <Frame.Background>
            <LinearGradientBrush EndPoint="0,1">
                <GradientStop Color="White"  Offset="0.9" />
                <GradientStop Color="LightBlue" Offset="1.0" />
            </LinearGradientBrush>
        </Frame.Background>

        <Label      Text="Report Fire"
                    FontSize="26"
                    HorizontalOptions="Center"
                    VerticalOptions="Center"
                    TextColor="#282828"
                    FontAttributes="Bold">
        </Label>
    </Frame>
</Frame>

<!--Time Found Label-->
<FlexLayout     Grid.Row="3"
                Grid.Column="1"
                BackgroundColor="#2daff0"
                HorizontalOptions="Center">

    <Label      Text="   Time Found @   "
                BackgroundColor="#2daFF0"
                FontSize="18"
                TextColor="black">
        <Label.Background>
            <LinearGradientBrush EndPoint="1,0">
                <GradientStop Color="LightBlue"  Offset="0.8" />
                <GradientStop Color="#2daff0" Offset="1.0" />
            </LinearGradientBrush>
        </Label.Background>
    </Label>
    <Label      x:Name="timeFound"
                TextColor="Black"
                FontSize="16"
                VerticalTextAlignment="Center"
                BackgroundColor="#2daff0">
    </Label>

</FlexLayout>

<!--Place Label-->
```

```
                    <FlexLayout     Grid.Row="4"
                                    Grid.Column="1"
                                    BackgroundColor="#2daff0"
                                    HorizontalOptions="Center">
                        <Label Text="   Place Name   "
                               BackgroundColor="#2daFF0"
                               FontSize="18"
                               TextColor="black">
                            <Label.Background>
                                <LinearGradientBrush EndPoint="1,0">
                                    <GradientStop Color="LightBlue"  Offset="0.8" />
                                    <GradientStop Color="#2daff0" Offset="1.0" />
                                </LinearGradientBrush>
                            </Label.Background>
                        </Label>
                        <Label  x:Name="placeName"
                                TextColor="Black"
                                FontSize="16"
                                VerticalTextAlignment="Center"
                                HorizontalTextAlignment="Center"
                                Margin="18,0,0,0"
                                VerticalOptions="Center"
                                BackgroundColor="#2daff0">
                        </Label>
                    </FlexLayout>

                    <!--Wind Direction // Start-->
                    <Label      Text="Wind Direction"
                                Grid.Row="6"
                                Grid.Column="1"
                                TextColor="Black"
                                FontSize="18">
                    </Label>
                    <Picker     x:Name="directionEntry"
                                Title="--Select--"
                                Grid.Row="7"
                                TextColor="Black"
                                Grid.Column="1"
                                FontSize="18"
                                BackgroundColor="LightGray"

SelectedIndexChanged="directionEntry_SelectedIndexChanged">
                    </Picker>
                    <!--Wind Direction // End-->

                    <!-- Latitude & Longitude // Start-->
                    <Label      x:Name="myLat"
                                Grid.Row="0"
                                Grid.Column="0"
                                TextColor="Black"
                                FontSize="2">
                    </Label>
```

```
<Label      x:Name="myLong"
            Grid.Row="0"
            Grid.Column="0"
            TextColor="Black"
            FontSize="2">
</Label>
<!-- Latitude & Longitude // End-->

<Label      x:Name="deviceID"
            Grid.Row="0"
            Grid.Column="0"
            TextColor="Black"
            FontSize="1">
</Label>

<!-- Description // Start-->
<Label      x:Name="FireDescription"
            Text="Fire Description"
            Grid.Row="9"
            Grid.Column="1"
            TextColor="Black"
            FontSize="18">
</Label>
<Editor     x:Name="FireDesc"
            Placeholder="Quick Description"
            MaxLength="100"
            HeightRequest="100"
            Grid.Row="10"
            FontSize="18"
            TextColor="Black"
            BackgroundColor="LightGray"
            Grid.Column="1"
            Keyboard="Text">
</Editor>
<!-- Description // End-->


<Entry      x:Name="fireID"
            Grid.Row="0"
            Grid.Column="0"
            FontSize="1"
            Keyboard="Numeric">
</Entry>

<!-- Photo // Start-->
<Button     x:Name="photo_Add"
            Grid.Row="11"
            Grid.Column="1"
            FontSize="20"
            Text="Add Photo"
            BackgroundColor="LightGray"
            WidthRequest="300"
            BorderColor="Black"
            TextColor="Black"
            HorizontalOptions="Center"
            CornerRadius="10"
            BorderWidth="3"
            ImageSource="add.png"
            ContentLayout="Right"
            Padding="75,0,40,0"
            FontAttributes="Bold"
            Clicked="photo_Add_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Photo Take Button-->
<Button     x:Name="photo_Take"
            Grid.Row="12"
            Grid.Column="1"
            FontSize="20"
            Text="Take Photo"
            BackgroundColor="LightGray"
            WidthRequest="300"
            BorderColor="Black"
            TextColor="Black"
            HorizontalOptions="Center"
            CornerRadius="10"
            BorderWidth="3"
            ImageSource="take.png"
            ContentLayout="Right"
            Padding="75,0,40,0"
            FontAttributes="Bold"
            Clicked="photo_Take_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Image Import-->
<Image      x:Name="imgChoose"
            HeightRequest="150"
            Grid.Row="13"
            Grid.Column="1"
            Source="photo.png">
</Image>
<!-- Photo // End-->
```

```
        <!--Report Fire Button-->
        <Button      WidthRequest="300"
                     BorderColor="Black"
                     TextColor="Black"
                     HorizontalOptions="Center"
                     Text="Report Fire"
                     CornerRadius="10"
                     BorderWidth="3"
                     FontSize="20"
                     ImageSource="flame.png"
                     ContentLayout="Right"
                     Padding="75,0,40,0"
                     FontAttributes="Bold"
                     Grid.Row="15"
                     Grid.Column="1"
                     Clicked="btn_Add_Clicked"
                     BackgroundColor="#ff8c19">
            <Button.Background>
                <LinearGradientBrush EndPoint="0,1">
                    <GradientStop Color="#ff8c19"  Offset="0.7" />
                    <GradientStop Color="White" Offset="1.0" />
                </LinearGradientBrush>
            </Button.Background>
        </Button>

        <!--Cancel Button-->
        <Button      Text="Cancel"
                     x:Name="btn_Cancel"
                     Grid.Row="16"
                     Grid.Column="1"
                     WidthRequest="300"
                     BorderColor="Black"
                     TextColor="Black"
                     HorizontalOptions="Center"
                     BackgroundColor="LightGray"
                     CornerRadius="10"
                     BorderWidth="3"
                     FontSize="20"
                     ImageSource="backarrow.png"
                     ContentLayout="Left"
                     Padding="40,0,75,0"
                     FontAttributes="Bold"
                     Clicked="btn_Cancel_Clicked">
            <Button.Background>
                <LinearGradientBrush EndPoint="0,1">
                    <GradientStop Color="LightGray"  Offset="0.1" />
                    <GradientStop Color="white" Offset="1.0" />
                </LinearGradientBrush>
            </Button.Background>
        </Button>
    </Grid>
```

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

Figure 63 - ReportFireInfoView.xaml

5.1.5.28. ReportFireInfoView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   ReportFireInfoView
* Purpose:     Backend for Report Fire.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Firebase.Database;
using Firebase.Database.Query;
using Wildfire.Helper;
using Wildfire.Models;
using Xamarin.Forms.Xaml;
using Xamarin.Essentials;
using Xamarin.Forms.GoogleMaps;
using Wildfire.ViewModels;
using Firebase.Storage;
using Plugin.Media;
using Plugin.Media.Abstractions;
using System.Diagnostics;
using System.IO;


namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class ReportFireInfoView : ContentPage
   {
       MediaFile file;
       MediaFile MediaFile;
       FirebaseHelper firebaseHelper = new FirebaseHelper();
       public static string recentFire;
       public ReportFireInfoView(double Lat, double Long, string Place)
       {
           InitializeComponent();

           myLat.Text = $"{Lat}";
           myLong.Text = $"{Long}";
           placeName.Text = $"{Place}";
```

```
            timeFound.Text = DateTime.Now.ToString("dd/MM/yyyy HH:mm");
            directionEntry.Items.Add("North");
            directionEntry.Items.Add("South");
            directionEntry.Items.Add("East");
            directionEntry.Items.Add("West");
            directionEntry.Items.Add("North-East");
            directionEntry.Items.Add("North-West");
            directionEntry.Items.Add("South-East");
            directionEntry.Items.Add("South-West");
            directionEntry.Items.Add("Unknown");

            // deviceID.Text = Android.OS.Build.GetSerial().ToString(); Optimal
solution

            deviceID.Text = Android.OS.Build.Fingerprint;
        }

        // Report Fire Event Handler
        private async void btn_Add_Clicked(object sender, EventArgs e)
        {
            if (directionEntry.SelectedIndex == -1)
            {
                await DisplayAlert("Error", "Please Select Wind Direction", "Yes");
                return;
            }
            if (FireDesc.Text == null)
            {
                await DisplayAlert("Error", "Please enter a description to help the
firefigter ", "Yes");
                return;
            }
            if (FireDesc.Text == "")
            {
                await DisplayAlert("Error", "Please enter a description to help the
firefigter ", "Yes");
                return;
            }
            else
            {

            }
            await firebaseHelper.AddFire(Convert.ToString(Id), myLat.Text,
myLong.Text, timeFound.Text, directionEntry.Items[directionEntry.SelectedIndex],
FireDesc.Text, placeName.Text, deviceID.Text);
            fireID.Text = string.Empty;
            myLat.Text = string.Empty;
            myLong.Text = string.Empty;
            timeFound.Text = string.Empty;
            directionEntry.Items[directionEntry.SelectedIndex] = string.Empty;
            FireDesc.Text = string.Empty;
            placeName.Text = string.Empty;
            deviceID.Text = string.Empty;
```

```
                MapView.reportedIndicator = 1;
                MapView.locationCount = 0;
                await DisplayAlert("Success", "Added", "OK");
                var allFires = await firebaseHelper.GetAllFires();
                await Navigation.PushModalAsync(new MainTabPage());
            }

            // Cancel Button Event Handler
            private async void btn_Cancel_Clicked(object sender, EventArgs e)
            {
                try
                {
                    MapView.reportedIndicator = 2;
                    await Navigation.PopModalAsync();
                }
                catch (Exception ex)
                {
                    ex.Message.ToString();
                }
            }

            // Picker Index Event Handler
            private void directionEntry_SelectedIndexChanged(object sender, EventArgs e)
            {
                try
                {
                    var name = directionEntry.Items[directionEntry.SelectedIndex];
                }
                catch (Exception ex)
                {
                    ex.Message.ToString();
                }
            }

            // Photo Add Event Handler
            private async void photo_Add_Clicked(object sender, EventArgs e)
            {
                await CrossMedia.Current.Initialize();
                try
                {
                    file = await Plugin.Media.CrossMedia.Current.PickPhotoAsync(new
    Plugin.Media.Abstractions.PickMediaOptions
                    {
                        PhotoSize = Plugin.Media.Abstractions.PhotoSize.Medium
                    });
                    if (file == null)
                        return;
                    imgChoose.Source = ImageSource.FromStream(() =>
                    {
                        var imageStram = file.GetStream();
                        return imageStram;
                    });
                await StoreImages(file.GetStream());
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex.Message);
            }
           await StoreImages(file.GetStream());
        }

        // Photo Take Event Handler
        private async void photo_Take_Clicked(object sender, EventArgs e)
        {
            await CrossMedia.Current.Initialize();
            try
            {
                MediaFile = await Plugin.Media.CrossMedia.Current.TakePhotoAsync(new
StoreCameraMediaOptions
                {
                    PhotoSize = PhotoSize.Medium,
                    AllowCropping = true
                });

                imgChoose.Source = ImageSource.FromStream(() =>
                {
                    var imageStram = MediaFile.GetStream();
                    return imageStram;
                });
                await StoreImages(MediaFile.GetStream());
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex.Message);
            }
        }

        // Store Image Task
        public async Task<string> StoreImages(Stream imageStream)
        {
            var fileName = Id;
            var stroageImage = await new
FirebaseStorage("driven-bulwark-297919.appspot.com")
                .Child("Fires")
                .Child(fileName + ".jpeg")

                .PutAsync(imageStream);
            string imgurl = stroageImage;
            return imgurl;
        }
    }
}
```

Figure 64 - ReportFireInfoView.xaml.cs.

5.1.5.29. ResolveFireInfoView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     ResolveFireInfoView
    Purpose:       Resolve fire interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.ResolveFireInfoView">

   <!--Background-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="White" Offset="0.1" />
           <GradientStop Color="#F7D9BB"  Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>
       <ScrollView>
           <Grid RowSpacing="10">
               <Grid.RowDefinitions>
                   <RowDefinition Height="0"></RowDefinition>
                   <RowDefinition Height="70"></RowDefinition>
                   <RowDefinition Height="25"></RowDefinition>
                   <RowDefinition Height="25"></RowDefinition>
                   <RowDefinition Height="100"></RowDefinition>
                   <RowDefinition Height="25"></RowDefinition>
                   <RowDefinition Height="80"></RowDefinition>
                   <RowDefinition Height="0"></RowDefinition>
                   <RowDefinition Height="56"></RowDefinition>
                   <RowDefinition Height="56"></RowDefinition>
                   <RowDefinition Height="100"></RowDefinition>
                   <RowDefinition Height="0"></RowDefinition>
                   <RowDefinition Height="56"></RowDefinition>
                   <RowDefinition Height="56"></RowDefinition>
               </Grid.RowDefinitions>
               <Grid.ColumnDefinitions>
                   <ColumnDefinition Width="*"/>
                   <ColumnDefinition Width="300"/>
                   <ColumnDefinition Width="*"/>
               </Grid.ColumnDefinitions>

                   <!--Header-->
                   <Frame      BorderColor="LightGray"
                               Padding="7"
                               HeightRequest="80"
                               WidthRequest="300"
```

```
                Grid.Row="1"
                Grid.Column="1"
                BackgroundColor="LightGray"
                HasShadow="True"
                CornerRadius="15"
                IsVisible="true"
                HorizontalOptions="Center">

        <Frame      BorderColor="LightGray"
                    Padding="7"
                    HeightRequest="80"
                    WidthRequest="310"
                    Grid.Row="1"
                    Grid.Column="1"
                    BackgroundColor="LightGray"
                    HasShadow="True"
                    CornerRadius="15"
                    IsVisible="true"
                    HorizontalOptions="Center">
            <Frame.Background>
                <LinearGradientBrush EndPoint="0,1">
                    <GradientStop Color="White"  Offset="0.9" />
                    <GradientStop Color="LightBlue" Offset="1.0" />
                </LinearGradientBrush>
            </Frame.Background>

            <Label      Text="Resolve Fire"
                        FontSize="26"
                        HorizontalOptions="Center"
                        VerticalOptions="Center"
                        TextColor="#282828"
                        FontAttributes="Bold">
            </Label>
        </Frame>
    </Frame>

<!--Fire Tag-->
<Label      x:Name="fireTag"
            Grid.Row="0"
            Grid.Column="0"
            FontSize="0">
</Label>

<!--Placename Label-->
<Label      x:Name="firePlaceName"
            Grid.Row="2"
            TextColor="Black"
            FontAttributes="Bold"
            HorizontalOptions="CenterAndExpand"
            Grid.Column="1"
            FontSize="20">
```

```
</Label>

<!--Fire ID-->
<Label      x:Name="fireID"
            Grid.Row="3"
            Grid.Column="1"
            FontSize="20"
            FontAttributes="Bold"
            HorizontalOptions="CenterAndExpand"
            TextColor="Black">
</Label>

<!--Image Import-->
<Image      x:Name="imgChoose"
            HeightRequest="150"
            Grid.Row="4"
            Grid.Column="1"
            Source="photo.png">
</Image>

<!--Activity Indicator-->
<ActivityIndicator      x:Name="overlay"
                        IsRunning="True"
                        IsVisible="True"
                        Color="Black"
                        BackgroundColor="Transparent"
                        Grid.Column="1"
                        Grid.Row="4">
</ActivityIndicator>

<!--Time Label-->
<Label      x:Name="time"
            Grid.Row="0"
            Grid.Column="0"
            TextColor="white"
            FontSize="0">
</Label>

<!--Fire Description-->
<Label      x:Name="FireDescription"
            Text="Fire Description"
            Grid.Row="5"
            Grid.Column="1"
            TextColor="Black"
            FontSize="18">
</Label>

<!--New description Label-->
<Editor     x:Name="newDesc"
            Placeholder=" Information about Resolved Fire"
            MaxLength="100"
            HeightRequest="100"
            BackgroundColor="LightGray"
            Grid.Row="6"
            Grid.Column="1"
            TextColor="Black"
            Keyboard="Text">
</Editor>

<!-- Add Photo Button-->
<Button     x:Name="photo_Add"
            Grid.Row="8"
            Grid.Column="1"
            FontSize="20"
            Text="Add Photo"
            BackgroundColor="LightGray"
            WidthRequest="300"
            BorderColor="Black"
            TextColor="Black"
            HorizontalOptions="Center"
            CornerRadius="10"
            BorderWidth="3"
            ImageSource="add.png"
            ContentLayout="Right"
            Padding="75,0,40,0"
            FontAttributes="Bold"
            Clicked="photo_Add_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Photo Take Button-->
<Button     x:Name="photo_Take"
            Grid.Row="9"
            Grid.Column="1"
            FontSize="20"
            Text="Take Photo"
            BackgroundColor="LightGray"
            WidthRequest="300"
            BorderColor="Black"
            TextColor="Black"
            HorizontalOptions="Center"
            CornerRadius="10"
            ImageSource="take.png"
            ContentLayout="Right"
            Padding="75,0,40,0"
            BorderWidth="3"
            FontAttributes="Bold"
            Clicked="photo_Take_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--New Image -->
<Image      x:Name="imgNewChoose"
            HeightRequest="150"
            Grid.Row="10"
            Grid.Column="1"
            Source="photo.png">
</Image>

<!--Resolve Button-->
<Button     Text="Resolve"
            x:Name="btn_Res"
            Grid.Row="12"
            Grid.Column="1"
            BackgroundColor="#ff8c19"
            FontSize="20"
            FontAttributes="Bold"
            CornerRadius="10"
            BorderWidth="3"
            BorderColor="Black"
            TextColor="Black"
            ImageSource="smallres.png"
            ContentLayout="Right"
            Padding="75,0,40,0"
            Clicked="btn_Res_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.7" />
            <GradientStop Color="White" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Cancel Button-->
<Button     Text="Cancel"
            x:Name="btn_Cancel"
            Grid.Row="13"
            Grid.Column="1"
            BackgroundColor="LightGray"
            FontSize="20"
            FontAttributes="Bold"
            CornerRadius="10"
            BorderWidth="3"
```

```
                                   ImageSource="backarrow.png"
                                   ContentLayout="Left"
                                   Padding="40,0,75,0"
                                   BorderColor="Black"
                                   TextColor="Black"
                                   Clicked="btn_Cancel_Clicked">
                               <Button.Background>
                                   <LinearGradientBrush EndPoint="0,1">
                                       <GradientStop Color="LightGray"  Offset="0.1" />
                                       <GradientStop Color="white" Offset="1.0" />
                                   </LinearGradientBrush>
                               </Button.Background>
                           </Button>
                       </Grid>
                   </ScrollView>
               </ContentPage.Content>
</ContentPage>
```

Figure 65 - ResolveFireInfoView.xaml

5.1.5.30. ResolveFireInfoView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:    ResolveFireInfoView
* Purpose:      Backend for Resolve Fire.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Firebase.Database;
using Firebase.Database.Query;
using Wildfire.Helper;
using Wildfire.Models;
using Xamarin.Forms.Xaml;
using Xamarin.Essentials;
using System.IO;
using Firebase.Storage;
using System.Net;
using Plugin.Media;
using Plugin.Media.Abstractions;
using System.Diagnostics;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
```

```
    public partial class ResolveFireInfoView : ContentPage
    {
        MediaFile file;
        MediaFile mediaFile;
        FirebaseHelper firebaseHelper = new FirebaseHelper();

        public ResolveFireInfoView(string Label, string placeName, string tag)
        {
            InitializeComponent();
            fireID.Text = $"{placeName}";
            firePlaceName.Text = $"{Label}";
            fireTag.Text = $"{tag}";
            time.Text = DateTime.Now.ToString("dd/MM/yyyy HH:mm");
            var filename = fireTag.Text;
        }

        /// <summary>
        /// Loading Logic
        /// </summary>
        protected async override void OnAppearing()
        {
            base.OnAppearing();
            overlay.IsVisible = true;
            imgChoose.IsVisible = false;
            await LoadImage();
            imgChoose.IsVisible = true;
            overlay.IsVisible = false;
        }

        /// <summary>
        /// Task to Load an image frm firebase
        /// </summary>
        public async Task LoadImage()
        {
            try
            {
                var filename = fireTag.Text;
                var webClient = new WebClient();
                var storageImage = await new
FirebaseStorage("driven-bulwark-297919.appspot.com")
                    .Child("Fires")
                    .Child(filename + ".jpeg")
                    .GetDownloadUrlAsync();
                string imgurl = storageImage;
                byte[] imgbytes = webClient.DownloadData(imgurl);
                imgChoose.Source = ImageSource.FromStream(() => new
MemoryStream(imgbytes));
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Resolve Button Event Handler
        private async void btn_Res_Clicked(object sender, EventArgs e)
        {
            if (newDesc.Text == null)
            {
                await DisplayAlert("Error", "Please enter a description to resolve
the fire ", "Yes");
                return;
            }
            if (newDesc.Text == "")
            {
                await DisplayAlert("Error", "Please enter a description to resolve
the fire ", "Yes");
                return;
            }
            MapView.locationCount = 0;
            await firebaseHelper.ResolveFire(Convert.ToString(fireTag.Text));
            await DisplayAlert("Success", "Fire Resolved", "OK");
            await firebaseHelper.AddResolvedFire(fireTag.Text, firePlaceName.Text,
fireID.Text, newDesc.Text,time.Text);
            await Navigation.PushModalAsync(new MainTabPage());
        }

        // Cancel Button Event Handler
        private async void btn_Cancel_Clicked(object sender, EventArgs e)
        {
            try
            {
                MapView.locationCount = 0;
                await Navigation.PushModalAsync(new MainTabPage());
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Add Photo Event Handler
        private async void photo_Add_Clicked(object sender, EventArgs e)
        {
            await CrossMedia.Current.Initialize();
            try
            {
                file = await Plugin.Media.CrossMedia.Current.PickPhotoAsync(new
Plugin.Media.Abstractions.PickMediaOptions
                {
                    PhotoSize = Plugin.Media.Abstractions.PhotoSize.Medium
                });
                if (file == null)
                    return;
                imgNewChoose.Source = ImageSource.FromStream(() =>
                {
                    var imageStram = file.GetStream();
                    return imageStram;
                });
                await StoreImages(file.GetStream());
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex.Message);
            }
            await StoreImages(file.GetStream());
        }

        // Store Images Task
        public async Task<string> StoreImages(Stream imageStream)
        {
            var fileName = fireTag.Text;
            var stroageImage = await new
FirebaseStorage("driven-bulwark-297919.appspot.com")
                .Child("Fires")
                .Child(fileName + "(new)" + ".jpeg")

                .PutAsync(imageStream);
            string imgurl = stroageImage;
            return imgurl;
        }

        // Take Photo Event Handler
        private async void photo_Take_Clicked(object sender, EventArgs e)
        {
            await CrossMedia.Current.Initialize();
            try
            {
                mediaFile = await
Plugin.Media.CrossMedia.Current.TakePhotoAsync(new StoreCameraMediaOptions
                {
                    PhotoSize = PhotoSize.Medium,
                    AllowCropping = true
                });

                imgNewChoose.Source = ImageSource.FromStream(() =>
                {
                    var imageStram = mediaFile.GetStream();
                    return imageStram;
                });
                await StoreImages(mediaFile.GetStream());
            }
            catch (Exception ex)
```

```
                {
                    Debug.WriteLine(ex.Message);
                }
            }
        }
    }
```

Figure 66 - ResolveFireInfoView.xaml.cs

5.1.5.31. SearchView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.SearchView"
            FocusOriginCommand="{Binding FocusOriginCommand}">


   <Grid VerticalOptions="FillAndExpand"
             BackgroundColor="White"
             RowSpacing="0"
             ColumnSpacing="0">
       <Grid.RowDefinitions>
           <RowDefinition Height="Auto"/>
           <RowDefinition Height="Auto"/>
           <RowDefinition Height="Auto"/>
       </Grid.RowDefinitions>
       <Grid.ColumnDefinitions>
           <ColumnDefinition Width="*"/>
       </Grid.ColumnDefinitions>

       <StackLayout    Grid.Row="0"
                       Padding="10"
                       BackgroundColor="White"
                       Orientation="Horizontal">
           <Image      Source="search.png"
                       HeightRequest="20"
                       WidthRequest="20"/>
           <Entry      Placeholder="Search a Location"
                       Text="{Binding PickupText}"
                       FontSize="18"
                       x:Name="originEntry"
                       ClassId="origin"
                       FontAttributes="Italic"
                       ReturnType="Search"
                       HorizontalOptions="FillAndExpand" />
           <Button     ImageSource="search.png"
                       Clicked="searchPlace_Clicked"/>
       </StackLayout>
```

```
<ListView       VerticalOptions="FillAndExpand"
                BackgroundColor="Transparent"
                Grid.Row="2"
                Footer=""
                SelectedItem="{Binding PlaceSelected, Mode=TwoWay}"
                x:Name="list"
                ItemsSource="{Binding Places}"
                HasUnevenRows="true"
                SeparatorColor="Gray">

    <ListView.Triggers>
        <DataTrigger    TargetType="ListView"
                         Binding="{Binding ShowRecentPlaces}"
                         Value="True">
            <Setter     Property="ItemsSource"
                        Value="{Binding RecentPlaces}" />
        </DataTrigger>
        <DataTrigger    TargetType="ListView"
                        Binding="{Binding ShowRecentPlaces}"
                        Value="False">
            <Setter     Property="ItemsSource"
                        Value="{Binding Places}" />
        </DataTrigger>
    </ListView.Triggers>
    <ListView.Header>
        <StackLayout    x:Name="recentSearchText"
                        IsVisible="{Binding ShowRecentPlaces}">
            <Label      LineBreakMode="WordWrap"
                        FontAttributes="Bold"
                        Margin="20,10"
                        x:Name="recentSearch"
                        Text="History"/>
        </StackLayout>
    </ListView.Header>
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <Grid Padding="15"
                              RowSpacing="2"
                              ColumnSpacing="15">

                    <Grid.RowDefinitions>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                    </Grid.RowDefinitions>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="Auto"/>
                        <ColumnDefinition Width="*"/>
                    </Grid.ColumnDefinitions>
```

```
                                <Image Source="pin.png"
                                                 HeightRequest="20"
                                                 WidthRequest="20"
                                                 VerticalOptions="Start"
                                                 Grid.Row="0"
                                                 Grid.Column="0"
                                                 Grid.RowSpan="2"/>

                                <Label LineBreakMode="MiddleTruncation"
                                                  Text="{Binding
StructuredFormatting.MainText}"
                                                  Grid.Row="0"
                                                  Grid.Column="1"/>

                                <Label LineBreakMode="MiddleTruncation"
                                                    Text="{Binding
StructuredFormatting.SecondaryText}"
                                                    TextColor="Gray"
                                                    Grid.Row="1"
                                                    Grid.Column="1"/>
                            </Grid>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </Grid>
    </ContentPage>
```

Figure 67 - SearchView.xaml

5.1.5.32. Search View.xaml.cs

```
using System.Windows.Input;
using Xamarin.Forms;
using System;
using Xamarin.Essentials;
using System.Linq;
using Xamarin.Forms.GoogleMaps;

using System.Collections.Generic;

using System.Text;
using System.Threading.Tasks;


using Xamarin.Forms.Xaml;
using Wildfire.Views;
using Wildfire.Helper;
using Xamarin.Forms.Internals;
using Wildfire.Services;

namespace Wildfire.Views
{
   public partial class SearchView : ContentPage
   {
       public static readonly BindableProperty FocusOriginCommandProperty =
          BindableProperty.Create(nameof(FocusOriginCommand), typeof(ICommand),
typeof(SearchView), null, BindingMode.TwoWay);

       public ICommand FocusOriginCommand
       {
           get { return (ICommand)GetValue(FocusOriginCommandProperty); }
           set { SetValue(FocusOriginCommandProperty, value); }
       }

       public SearchView()
       {
           InitializeComponent();

       }

       protected override void OnBindingContextChanged()
       {
           base.OnBindingContextChanged();
           if (BindingContext != null)
           {
               FocusOriginCommand = new Command(OnOriginFocus);
           }
       }

       void OnOriginFocus()
       {
           originEntry.Focus();
       }

       private async void searchPlace_Clicked(object sender, EventArgs e)
       {

           var search = originEntry.Text;
           var searchLocation = await Geocoding.GetLocationsAsync(search);
           var sourceLocations = searchLocation?.FirstOrDefault();
           Location sourceCoordinates = new Location(sourceLocations.Latitude,
sourceLocations.Longitude);




           await Navigation.PushModalAsync(new MainTabPage());
```

```
            }
        }
    }
```

Figure 68 - SearchView.xaml.cs

5.1.5.33. SettingsView.xaml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     SettingsView
    Purpose:       Settings View interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Wildfire.Views.SettingsView"
             Title="Settings">

   <!--Background Color-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="#ff8c19"  Offset="0.0002" />
           <GradientStop Color="White" Offset="0.25" />
           <GradientStop Color="White" Offset="0.55" />
           <GradientStop Color="#F7D9BB" Offset="0.8" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Header-->
   <ContentPage.Content>
       <ScrollView>
       <Grid RowSpacing="7">
           <Grid.RowDefinitions>
               <RowDefinition Height="0"/>
               <RowDefinition Height="0"/>
               <RowDefinition Height="50"/>
               <RowDefinition Height="0"/>
               <RowDefinition Height="40" />
               <RowDefinition Height="100"/>
               <RowDefinition Height="20"/>
               <RowDefinition Height="45"/>
               <RowDefinition Height="35"/>
               <RowDefinition Height="0"/>
               <RowDefinition Height="56"/> <!-- 10 -->
               <RowDefinition Height="0"/>
               <RowDefinition Height="56"/>
               <RowDefinition Height="0"/>
               <RowDefinition Height="56"/>
               <RowDefinition Height="25"/>
           </Grid.RowDefinitions>
           <Grid.ColumnDefinitions>
               <ColumnDefinition Width="*"/>
               <ColumnDefinition Width="300"/>
               <ColumnDefinition Width="*"/>
           </Grid.ColumnDefinitions>

           <!--Day Label-->
           <Label      Grid.Row="2"
                       Grid.Column="1"
```

```
            Text="hello"
            x:Name="settingsDay"
            FontSize="34"
            FontAttributes="Bold"
            HorizontalOptions="Center"
            VerticalTextAlignment="Center"
            VerticalOptions="Center"
            TextColor="Black">
</Label>

<!--Date Label-->
<Label      Grid.Row="4"
            Grid.Column="1"
            Text=" hello"
            x:Name="settingsDate"
            FontSize="24"
            HorizontalOptions="Center"
            VerticalOptions="Center"
            TextColor="Black">
</Label>

<!--Logo Import-->
<Image      Source="xhdpi.png"
            Grid.Row="5"
            Grid.Column="1"
            HeightRequest="150"
            WidthRequest="150">
</Image>

<!--Notification On or Off Label-->
<Label      Grid.Row="8"
            Grid.Column="1"
            FontSize="18"
            TextColor="black"
            HorizontalOptions="Start"
            Text="Notification ON/OFF">
</Label>

<!--Notification Checkbox-->
<CheckBox   x:Name="notificationSettings"
            HorizontalOptions="End"
            Grid.Row="8"
            Grid.Column="1"
            IsChecked="{Binding NotificationEnabled}"
            CheckedChanged="OnChange">
</CheckBox>

<!--Nt=otification Radius-->
<Label      Grid.Row="7"
            Grid.Column="1"
            FontSize="18"
            TextColor="Black"
            HorizontalOptions="Start"
            VerticalOptions="Center"
            Text="Notification Radius">
</Label>

<!--Radius Entry-->
<Entry      x:Name="radiusSettings"
            TextColor="Black"
            PlaceholderColor="White"
            HorizontalTextAlignment="End"
            Grid.Row="7"
            WidthRequest="132"
            Grid.Column="1"
            BackgroundColor="LightGray"
            HorizontalOptions="End"
            Text="{Binding Radius}"
            Placeholder="Set Radius(Km)"
            Keyboard="Numeric"
            Completed="OnChange"
            ClearButtonVisibility="WhileEditing"
            ReturnType="Done">
</Entry>

<!--Change Password Button-->
<Button     Text="Change Password"
            Grid.Row="10"
            Grid.Column="1"
            FontSize="20"
            CornerRadius="10"
            BorderColor="Black"
            BorderWidth="3"
            HorizontalOptions="StartAndExpand"
            WidthRequest="300"
            BackgroundColor="LightCyan"
            TextColor="Black"
            FontAttributes="bold"
            x:Name="ChangePass"
            Clicked="ChangePass_Clicked">
        <Button.Background>
            <LinearGradientBrush EndPoint="0,1">
                <GradientStop Color="LightCyan"  Offset="0.1" />
                <GradientStop Color="white" Offset="1.0" />
            </LinearGradientBrush>
        </Button.Background>
</Button>

<!--Login Button-->
<Button     Text="Login"
            Grid.Row="12"
            Grid.Column="1"
```

```
                Clicked="Login_Button_Clicked"
                FontSize="20"
                FontAttributes="Bold"
                CornerRadius="10"
                BorderWidth="3"
                BorderColor="Black"
                TextColor="Black"
                HorizontalOptions="Start"
                WidthRequest="145"
                BackgroundColor="#ff8c19">
            <Button.Background>
                <LinearGradientBrush EndPoint="0,1">
                    <GradientStop Color="#ff8c19"  Offset="0.6" />
                    <GradientStop Color="white" Offset="1.0" />
                </LinearGradientBrush>
            </Button.Background>
        </Button>

    <!--Sign Up Button-->
    <Button     Text="Signup"
                Grid.Row="12"
                Grid.Column="1"
                Clicked="Signup_Button_Clicked"
                FontSize="20"
                FontAttributes="Bold"
                CornerRadius="10"
                BorderWidth="3"
                BorderColor="Black"
                TextColor="Black"
                HorizontalOptions="End"
                WidthRequest="145"
                BackgroundColor="LightGray">
            <Button.Background>
                <LinearGradientBrush EndPoint="0,1">
                    <GradientStop Color="LightGray"  Offset="0.6" />
                    <GradientStop Color="white" Offset="1.0" />
                </LinearGradientBrush>
            </Button.Background>
        </Button>

    <!--Logout Button-->
    <Button     Text="Logout"
                Grid.Row="14"
                Grid.Column="1"
                Clicked="Logout_Button_Clicked"
                FontSize="20"
                FontAttributes="Bold"
                CornerRadius="10"
                BorderWidth="3"
                BorderColor="Black"
                TextColor="Black"
```

```
                            ImageSource="enter.png"
                            ContentLayout="Right"
                            Padding="75,0,40,0"
                            HorizontalOptions="CenterAndExpand"
                            WidthRequest="300"
                            BackgroundColor="#2daff0">
                        <Button.Background>
                            <LinearGradientBrush EndPoint="0,1">
                                <GradientStop Color="#2daff0"  Offset="0.6" />
                                <GradientStop Color="white" Offset="1.0" />
                            </LinearGradientBrush>
                        </Button.Background>
                    </Button>
                </Grid>
            </ScrollView>
        </ContentPage.Content>
</ContentPage>
```

Figure 69 - SettingsView.xaml

5.1.5.34. SettingsView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   SettingsView
* Purpose:     Backend for Settings View.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using Xamarin.Essentials;

namespace Wildfire.Views
{

   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class SettingsView : ContentPage
   {
       IAuth auth;
       public static string radius;
       public static bool isChecked;
       public SettingsView()
       {
           InitializeComponent();
           radiusSettings.Text = Preferences.Get("radiusSettings", string.Empty);
```

```
            radius = radiusSettings.Text;
            if (radius == "")
            {
                radius =null;
            }
            notificationSettings.IsChecked =
Preferences.Get("notificationSettings", false);

            OnPropertyChanged();

            auth = DependencyService.Get<IAuth>();
            settingsDay.Text = DateTime.Now.DayOfWeek.ToString();
            settingsDate.Text = DateTime.Now.Date.ToString("dd/MM/yyyy");
        }

        protected override bool OnBackButtonPressed()
        {
            return true;
        }

        // Login Button Event Handler
        private async void Login_Button_Clicked(object sender, EventArgs e)
        {
            if (LoginPageView.token == null)
            {
                MapView.notificationCount = 1;
                await Navigation.PushModalAsync(new LoginPageView());
            }
            else
            {
                await DisplayAlert("Already Logged in", "", "Yes");
            }
        }

        // SignUp Button Event Handler
        private async void Signup_Button_Clicked(object sender, EventArgs e)
        {
            if (LoginPageView.token != null)
            {
                await DisplayAlert("You already have an account", "", "Yes");
            }
            else
            {
                await Navigation.PushModalAsync(new SignUpPageView());
            }
        }

        // Logout Button Event Handler
        private async  void Logout_Button_Clicked(object sender, EventArgs e)
        {
            var signOut = auth.SignOut();
            if (LoginPageView.token != null)
            {
                if (signOut)
                {
                    MapView.notificationCount = 0;
                    LoginPageView.token = null;
                    MapView.fireNotCount = 0;
                    MapView.loginCount = 0;
                    await Navigation.PushModalAsync(new FirstPageView());
                }
            }
            else
            {
               await  DisplayAlert("Not Logged in", "", "Yes");
            }
        }

        // OnChange Event Handler
        private void OnChange(object sender, EventArgs e)
        {
            Preferences.Set("radiusSettings", radiusSettings.Text);
            Preferences.Set("notificationSettings",
notificationSettings.IsChecked);

            if (radiusSettings.Text == string.Empty)
            {
                radiusSettings.Text = null;
            }
            else if (radiusSettings.Text == "0")
            {
                radiusSettings.Text = null;
            }
            radius = radiusSettings.Text;
            isChecked = notificationSettings.IsChecked;
            MapView.locationCount = 0;
            MapView.notificationCount = 0;
            MapView.fireNotCount = 0;
        }

        // OnDisapperaing Event Handler
        protected override void OnDisappearing()
        {
            base.OnDisappearing();
        }

        // Change Password Button Event Handler
        private async void ChangePass_Clicked(object sender, EventArgs e)
        {
            if (LoginPageView.token != null)
            {
                await Navigation.PushModalAsync(new ChangePass());
            }
            else
            {
                await DisplayAlert("Error", "Please Login", "ok");
            }
        }
    }
}
```

Figure 70 - SettingsView.xaml.cs

5.1.5.35. SignUpView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     SignUpPageView
    Purpose:       sign up view interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.Views.SignUpPageView">

   <!--Background-->
   <ContentPage.Background>
       <LinearGradientBrush EndPoint="0,1">
           <GradientStop Color="White"  Offset="0.8" />
           <GradientStop Color="#ff8c19" Offset="1.0" />
       </LinearGradientBrush>
   </ContentPage.Background>

   <!--Content Page-->
   <ContentPage.Content>
       <ScrollView>
           <Grid RowSpacing="12">
               <Grid.RowDefinitions>
               <RowDefinition Height="135"></RowDefinition>
               <RowDefinition Height="100"></RowDefinition>
               <RowDefinition Height="10"></RowDefinition>
               <RowDefinition Height="50"></RowDefinition>
               <RowDefinition Height="0"></RowDefinition>
               <RowDefinition Height="50"></RowDefinition>
               <RowDefinition Height="0"></RowDefinition>
               <RowDefinition Height="45"></RowDefinition>
               <RowDefinition Height="0"></RowDefinition>
               <RowDefinition Height="56"></RowDefinition>
               <RowDefinition Height="56"></RowDefinition>
           </Grid.RowDefinitions>
```

```
<Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"></ColumnDefinition>
        <ColumnDefinition Width="300"></ColumnDefinition>
        <ColumnDefinition Width="*"></ColumnDefinition>
</Grid.ColumnDefinitions>

    <!--Create account label-->
    <Label      Text="Create an Account"
                HorizontalTextAlignment="Center"
                TextColor="Black"
                FontSize="50"
                Margin="0, 10"
                FontAttributes="Bold"
                Grid.Row="0"
                Grid.Column="1">
    </Label>

    <!--Logo Import-->
    <Image      Source="xhdpi.png"
                Grid.Row="1"
                HeightRequest="150"
                WidthRequest="150"
                Grid.Column="1">
    </Image>

    <!--Email Entry-->
    <Entry      Placeholder="E-mail"
                Keyboard="Email"
                x:Name="EmailInput"
                BackgroundColor="LightGray"
                PlaceholderColor="DarkGray"
                TextColor="Black"
                Grid.Row="3"
                WidthRequest="200"
                Grid.Column="1">
    </Entry>

    <!--Password Entry-->
    <Entry      Placeholder="Password"
                IsPassword="True"
                x:Name="PasswordInput"
                BackgroundColor="LightGray"
                TextColor="Black"
                PlaceholderColor="DarkGray"
                Grid.Row="5"
                WidthRequest="200"
                Grid.Column="1">
    </Entry>

    <!--Firefighter Label-->
    <Label      Text="Please confirm you are a verified Firefighter?"
```

```
            TextColor="Black"
            HorizontalOptions="CenterAndExpand"
            Grid.Row="7"
            Grid.Column="1"
            FontSize="18"
            LineBreakMode="CharacterWrap"
            Margin="25,0"
            HorizontalTextAlignment="Start">
</Label>

<!--Checkbox-->
<CheckBox   Grid.Row="7"
            Grid.Column="1"
            HorizontalOptions="End"
            Margin=" 30,0"
            x:Name ="firefighterCon">
</CheckBox>

<!--Sign Up Button-->
<Button     Text="Sign Up"
            Clicked="SignUp_Clicked"
            BackgroundColor="#ff8c19"
            TextColor="Black"
            BorderColor="Black"
            CornerRadius="10"
            BorderWidth="3"
            FontSize="20"
            FontAttributes="Bold"
            Grid.Row="9"
            Grid.Column="1">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.6" />
            <GradientStop Color="White" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Back Button-->
<Button     Text="Back"
            Clicked="Back_Clicked"
            BackgroundColor="LightGray"
            TextColor="black"
            BorderColor="Black"
            CornerRadius="10"
            BorderWidth="3"
            FontSize="20"
            FontAttributes="Bold"
            Grid.Row="10"
            ImageSource="backarrow.png"
            ContentLayout="Left"
```

```
                            Padding="40,0,75,0"
                            Grid.Column="1">
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0,1">
                            <GradientStop Color="LightGray"  Offset="0.1" />
                            <GradientStop Color="white" Offset="1.0" />
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
            </Grid>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Figure 71 - SignUpView.xaml

5.1.5.36 SignUpView.xaml.cs

```
/* Author:      Jack McNally
 * Page Name:   SignUpPageView
 * Purpose:     Backend for Sign Up View.
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class SignUpPageView : ContentPage
   {
       IAuth auth;
       public SignUpPageView()
       {
           InitializeComponent();
           auth = DependencyService.Get<IAuth>();
           MapView.loginCount = 0;
       }

       // Sign Up Button Event Handler
       async void SignUp_Clicked(object sender, EventArgs e)
       {
           if(EmailInput.Text == null)
```

```
            {
                await DisplayAlert("Error", "Please enter email", "Ok");
                return;
            }
            if(PasswordInput.Text == null)
            {
                await DisplayAlert("Error", "Please enter password", "Ok");
                return;
            }

            if (PasswordInput.Text.Length < 8)
            {
                await DisplayAlert("Error", "Password must be greater than 8
characters", "ok");
            }
            else if (firefighterCon.IsChecked != true)
            {
                await DisplayAlert("Error", "Please verify is you are a
firefighter", "ok");
            }
            else
            {
                var user = auth.SignUpWithEmailAndPassword(EmailInput.Text,
PasswordInput.Text);
                if (user != null)
                {
                    await DisplayAlert("Success", "Created", "Ok");
                    var signOut = auth.SignOut();

                    if (signOut)
                    {
                        await Navigation.PushModalAsync(new LoginPageView());
                    }
                    else
                    {
                        await DisplayAlert("Error", "unable to LogOut", "Ok");
                    }
                }
            }
        }

        // Back Button Event Handler
        private async void Back_Clicked(object sender, EventArgs e)
        {
            try
            {
                await Navigation.PopModalAsync();
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
                }
            }
        }
    }
```

Figure 72 - SignUpPageView.xaml.cs

5.1.5.37. SplashScreen.cs

```
/* Author:      Jack McNally
* Page Name:   Splash Screen
* Purpose:     Screen that appears while app opens.
*/
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace Wildfire.Views
{
   class SplashScreen : ContentPage
   {
       Image image;

       public SplashScreen()
       {
           var sub = new AbsoluteLayout();
           image = new Image
           {
               Source = "xhdpi.png",
               WidthRequest = 150,
               HeightRequest = 150
           };

           AbsoluteLayout.SetLayoutFlags(image,
               AbsoluteLayoutFlags.PositionProportional);
           AbsoluteLayout.SetLayoutBounds(image,
               new Rectangle(0.5, 0.5, -1, -1));

           sub.Children.Add(image);

           this.BackgroundColor = Color.FromHex("#ffffff");
           this.Content = sub;
       }
       /// <summary>
       /// Loading Logic
       /// </summary>
       protected override async void OnAppearing()
```

```
        {
            base.OnAppearing();
            await image.ScaleTo(1, 2000);
            await image.ScaleTo(0.9, 1500, Easing.Linear);
            await image.ScaleTo(150, 1200, Easing.Linear);
            await Navigation.PushModalAsync(new FirstPageView());
        }
    }
}
```

Figure 73 - SplashScreen.cs

5.1.5.38. WildlandInfoView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
     Page Name:     WildlandInfoView
     Purpose:       wildland fire interface.
-->
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Wildfire.Views.WildlandInfoView">

    <!--Background-->
    <ContentPage.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="#ff8c19"  Offset="0.1" />
            <GradientStop Color="#F7D9BB" Offset="1.0" />
        </LinearGradientBrush>
    </ContentPage.Background>

    <!--Content Page-->
    <ContentPage.Content>

        <AbsoluteLayout>

            <!--Header-->
            <Frame      BorderColor="LightGray"
                        Padding="7"
                        HeightRequest="80"
                        WidthRequest="300"
                        AbsoluteLayout.LayoutFlags="PositionProportional"
                        AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                        BackgroundColor="LightGray"
                        HasShadow="True"
            IsVisible="true"
            CornerRadius="15"
            x:Name="FrameLabel"
            HorizontalOptions="Center">

    <Frame      BorderColor="LightGray"
                Padding="7"
                HeightRequest="80"
                WidthRequest="310"
                AbsoluteLayout.LayoutFlags="PositionProportional"
                AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                BackgroundColor="LightGray"
                HasShadow="True"
                CornerRadius="15"
                IsVisible="true"
                x:Name="FrameLabel1"
                HorizontalOptions="Center">
        <Frame.Background>
            <LinearGradientBrush EndPoint="0,1">
                <GradientStop Color="White"  Offset="0.9" />
                <GradientStop Color="LightBlue" Offset="1.0" />
            </LinearGradientBrush>
        </Frame.Background>

        <Label      Text="Wildland Fire Safety"
                    FontSize="26"
                    HorizontalOptions="Center"
                    VerticalOptions="Center"
                    TextColor="#282828"
                    FontAttributes="Bold">
        </Label>
    </Frame>
</Frame>

<!--Button 1-->
<Button     x:Name="BeforeFireInfo"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.25,-1,-1"
            BackgroundColor="LightBlue"
            CornerRadius="20"
            FontSize="25"
            FontAttributes="Italic"
            BorderWidth="3"
            BorderColor="Black"
            Text="Before a Wildfire"
            TextColor="Black"
            HeightRequest="110"
            WidthRequest="270"
            Clicked="BeforeFireInfo_Clicked">
    <Button.Background>
```

```
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue" Offset="0.3" />
            <GradientStop Color="LightSalmon" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Button 2-->
<Button     x:Name="DuringFireInfo"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.45,-1,-1"
            BackgroundColor="LightBlue"
            CornerRadius="20"
            FontSize="25"
            FontAttributes="Italic"
            BorderWidth="3"
            BorderColor="Black"
            Text="During a Wildfire"
            TextColor="Black"
            HeightRequest="110"
            WidthRequest="270"
            Clicked="DuringFireInfo_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue" Offset="0.3" />
            <GradientStop Color="LightSalmon" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Button 3-->
<Button     x:Name="AfterFireInfo"
            AbsoluteLayout.LayoutFlags="PositionProportional"
            AbsoluteLayout.LayoutBounds="0.5,0.65,-1,-1"
            BackgroundColor="LightBlue"
            CornerRadius="20"
            FontSize="25"
            FontAttributes="Italic"
            BorderWidth="3"
            BorderColor="Black"
            Text="After a Wildfire"
            TextColor="Black"
            HeightRequest="110"
            WidthRequest="270"
            Clicked="AfterFireInfo_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="1,0">
            <GradientStop Color="LightBlue" Offset="0.3" />
            <GradientStop Color="LightSalmon" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!--Back Button-->
<Button      x:Name="BackButton"
             AbsoluteLayout.LayoutFlags="PositionProportional"
             AbsoluteLayout.LayoutBounds="0.5,0.85,-1,-1"
             BackgroundColor="LightGray"
             CornerRadius="10"
             FontSize="20"
             FontAttributes="Bold"
             BorderWidth="3"
             BorderColor="Black"
             Text="Back"
             TextColor="Black"
             HeightRequest="56"
             ImageSource="backarrow.png"
             ContentLayout="Left"
             Padding="40,0,75,0"
             WidthRequest="300"
             Clicked="BackButton_Clicked">
    <Button.Background>
        <LinearGradientBrush EndPoint="0,1">
            <GradientStop Color="LightGray"  Offset="0.1" />
            <GradientStop Color="white" Offset="1.0" />
        </LinearGradientBrush>
    </Button.Background>
</Button>

<!-- Before Fire -->
<Frame       HeightRequest="680"
             WidthRequest="310"
             AbsoluteLayout.LayoutFlags="PositionProportional"
             AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
             BackgroundColor="Transparent"
             x:Name="beforeInfo"
             IsVisible="false"
             HasShadow="True">

    <Frame BackgroundColor="#f2f2f2"
           CornerRadius="15"
           HasShadow="True">

        <StackLayout>

            <Image HorizontalOptions="End"
                   HeightRequest="20"
                   WidthRequest="20"
                   Source="cancel.png">
```

```
                            <Image.GestureRecognizers>
                                <TapGestureRecognizer
Tapped="BeforeRemovePopupTapped"
                                                      NumberOfTapsRequired="1">
                                </TapGestureRecognizer>
                            </Image.GestureRecognizers>

                        </Image>

                        <Frame      BorderColor="LightGray"
                                    Padding="7"
                                    HeightRequest="80"
                                    WidthRequest="300"

AbsoluteLayout.LayoutFlags="PositionProportional"
                                    AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                                    BackgroundColor="LightGray"
                                    HasShadow="True"
                                    CornerRadius="15"
                                    IsVisible="true"
                                    HorizontalOptions="Center">

                            <Frame      BorderColor="LightGray"
                                        Padding="7"
                                        HeightRequest="80"
                                        WidthRequest="310"

AbsoluteLayout.LayoutFlags="PositionProportional"

AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                                        BackgroundColor="LightGray"
                                        HasShadow="True"
                                        CornerRadius="15"
                                        IsVisible="true"
                                        HorizontalOptions="Center">
                                <Frame.Background>
                                    <LinearGradientBrush EndPoint="0,1">
                                        <GradientStop Color="White"
Offset="0.9" />

                                        <GradientStop Color="LightBlue"

Offset="1.0" />
                                    </LinearGradientBrush>
                                </Frame.Background>

                                <Label      Text="Before a Wildfire"
                                            FontSize="26"
                                            HorizontalOptions="Center"
                                            VerticalOptions="Center"
                                            TextColor="#282828"
                                            FontAttributes="Bold">
                                </Label>
                            </Frame>
                        </Frame>

                        <ScrollView>

                            <StackLayout>

                                <Label      Text="The best measure to keep your
home or business safe from a Wildfire is to plan ahead."
                                            TextColor="Black"
                                            FontSize="18"
                                            HorizontalTextAlignment="Center"
                                            WidthRequest="255"
                                            IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                            HorizontalOptions="Center"
                                            HeightRequest="1"
                                            WidthRequest="300"
                                            Color="#5b5d68">
                                </BoxView>

                                <Label      Text="Follow these tips to give
yourself the best chance"

                                            TextColor="Black"
                                            FontSize="19"
                                            FontAttributes="Bold"
                                            HorizontalTextAlignment="Center">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                            HorizontalOptions="Center"
                                            HeightRequest="1"
                                            WidthRequest="300"
                                            Color="#5b5d68">
                                </BoxView>

                                <Label      Text="- Clear combustible materials
"

                                            TextColor="DarkRed"
                                            FontSize="20"
                                            FontAttributes="Bold"
                                            IsVisible="true">
                                </Label>

                                <Label      Text="Clear materials such as leaves
or firewood away from the home."

                                            TextColor="Black"
```

```
                                    FontSize="16"
                                    FontAttributes="None"
                                    IsVisible="true">
                        </Label>

                        <BoxView    VerticalOptions="Center"
                                    HorizontalOptions="Center"
                                    HeightRequest="1"
                                    WidthRequest="300"
                                    Color="#5b5d68">
                        </BoxView>

                        <Label      Text="- Remove vegetation"
                                    TextColor="DarkRed"
                                    FontSize="20"
                                    FontAttributes="Bold"
                                    IsVisible="true">
                        </Label>

                        <Label      Text="Remove flowers, vines or any
vegetation that could catch fire."
                                    TextColor="Black"
                                    FontSize="16"
                                    FontAttributes="None"
                                    IsVisible="true">
                        </Label>

                        <BoxView    VerticalOptions="Center"
                                    HorizontalOptions="Center"
                                    HeightRequest="1"
                                    WidthRequest="300"
                                    Color="#5b5d68">
                        </BoxView>

                        <Label      Text="- Clear furniture."
                                    TextColor="DarkRed"
                                    FontSize="20"
                                    FontAttributes="Bold"
                                    IsVisible="true">
                        </Label>

                        <Label      Text="Place outdoor furniture such a
tables and chairs in a safe storage area."
                                    TextColor="Black"
                                    FontSize="16"
                                    FontAttributes="None"
                                    IsVisible="true">
                        </Label>

                        <BoxView    VerticalOptions="Center"
```

```
                                        HorizontalOptions="Center"
                                        HeightRequest="1"
                                        WidthRequest="300"
                                        Color="#5b5d68">
                                </BoxView>

                                <Label      Text="- Create fuel breaks"
                                            TextColor="DarkRed"
                                            FontSize="20"
                                            FontAttributes="Bold"
                                            IsVisible="true">
                                </Label>

                                <Label      Text="Fuel breaks are made from non
combustible materials and act as a barrier to stop a wildfire from getting to a
building."
                                            TextColor="Black"
                                            FontSize="16"
                                            FontAttributes="None"
                                            IsVisible="true">
                                </Label>



                            </StackLayout>
                        </ScrollView>
                    </StackLayout>
                </Frame>
            </Frame>

            <!-- During -->
            <Frame      HeightRequest="620"
                        WidthRequest="310"
                        AbsoluteLayout.LayoutFlags="PositionProportional"
                        AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
                        BackgroundColor="Transparent"
                        x:Name="duringInfo"
                        IsVisible="false"
                        HasShadow="True">

                <Frame BackgroundColor="#f2f2f2"
                       CornerRadius="15"
                       HasShadow="True">

                    <StackLayout>

                        <Image HorizontalOptions="End"
                               HeightRequest="20"
                               WidthRequest="20"
                               Source="cancel.png">
```

```
                                <Image.GestureRecognizers>
                                    <TapGestureRecognizer
Tapped="DuringRemovePopupTapped"
                                                          NumberOfTapsRequired="1">
                                    </TapGestureRecognizer>
                                </Image.GestureRecognizers>

                            </Image>

                            <Frame      BorderColor="LightGray"
                                        Padding="7"
                                        HeightRequest="80"
                                        WidthRequest="300"

AbsoluteLayout.LayoutFlags="PositionProportional"
                                        AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                                        BackgroundColor="LightGray"
                                        HasShadow="True"
                                        CornerRadius="15"
                                        IsVisible="true"
                                        HorizontalOptions="Center">

                                <Frame      BorderColor="LightGray"
                                            Padding="7"
                                            HeightRequest="80"
                                            WidthRequest="310"

AbsoluteLayout.LayoutFlags="PositionProportional"

AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                                            BackgroundColor="LightGray"
                                            HasShadow="True"
                                            CornerRadius="15"
                                            IsVisible="true"
                                            HorizontalOptions="Center">
                                    <Frame.Background>
                                        <LinearGradientBrush EndPoint="0,1">
                                            <GradientStop Color="White"
Offset="0.9" />

                                            <GradientStop Color="LightBlue"

Offset="1.0" />
                                        </LinearGradientBrush>
                                    </Frame.Background>

                                    <Label      Text="During a Wildfire"
                                                FontSize="26"
                                                HorizontalOptions="Center"
                                                VerticalOptions="Center"
                                                TextColor="#282828"
                                                FontAttributes="Bold">
                                    </Label>
                                </Frame>
                            </Frame>

                            <ScrollView>

                                <StackLayout>

                                    <Label      Text="If a wildfire has been
reported in your area, Activate your safety plan."
                                                TextColor="Black"
                                                FontSize="18"
                                                HorizontalTextAlignment="Center"
                                                IsVisible="true">
                                    </Label>

                                    <BoxView    VerticalOptions="Center"
                                                HorizontalOptions="Center"
                                                HeightRequest="1"
                                                WidthRequest="300"
                                                Color="#5b5d68">
                                    </BoxView>

                                    <Label      Text="Here are some tips to aid you
during a wildfire."

                                                TextColor="Black"
                                                FontSize="19"
                                                FontAttributes="Bold"
                                                HorizontalTextAlignment="Center"
                                                IsVisible="true">
                                    </Label>

                                    <BoxView    VerticalOptions="Center"
                                                HorizontalOptions="Center"
                                                HeightRequest="1"
                                                WidthRequest="300"
                                                Color="#5b5d68">
                                    </BoxView>

                                    <Label      Text="- Prepare to Evacuate"
                                                TextColor="DarkRed"
                                                FontSize="20"
                                                FontAttributes="Bold"
                                                IsVisible="true">
                                    </Label>

                                    <Label      Text="If a wildfire has been
reported it is important to listen to emergency advice, if the call to evacute
is made. You must be ready to evacuate as quickly and as safely as possible"
```

```
                                    TextColor="Black"
                                    FontSize="16"
                                    FontAttributes="None"
                                    IsVisible="true">
                        </Label>

                        <BoxView    VerticalOptions="Center"
                                    HorizontalOptions="Center"
                                    HeightRequest="1"
                                    WidthRequest="300"
                                    Color="#5b5d68">
                        </BoxView>

                        <Label      Text="- Evacuate"
                                    TextColor="DarkRed"
                                    FontSize="20"
                                    FontAttributes="Bold"
                                    IsVisible="true">
                        </Label>

                        <Label      Text="When evacuating it is
important to do so in the safest manner. Follow professional advice of where to
go, if you are in a vehicle keep vents closed and keep the door unlocked as in
the event of an emergency, locked door may slow rescue time."
                                    TextColor="Black"
                                    FontSize="16"
                                    FontAttributes="None"
                                    IsVisible="true">
                        </Label>


                    </StackLayout>
                </ScrollView>
            </StackLayout>
        </Frame>
    </Frame>

    <!-- After -->
    <Frame      HeightRequest="670"
                WidthRequest="310"
                AbsoluteLayout.LayoutFlags="PositionProportional"
                AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
                BackgroundColor="Transparent"
                x:Name="afterInfo"
                IsVisible="false"
                HasShadow="True">

        <Frame BackgroundColor="#f2f2f2"
               CornerRadius="15"
               HasShadow="True">
```

```
                    <StackLayout>

                        <Image HorizontalOptions="End"
                               HeightRequest="20"
                               WidthRequest="20"
                               Source="cancel.png">

                            <Image.GestureRecognizers>
                                <TapGestureRecognizer
Tapped="AfterRemovePopupTapped"
                                                      NumberOfTapsRequired="1">
                                </TapGestureRecognizer>
                            </Image.GestureRecognizers>

                        </Image>

                        <Frame      BorderColor="LightGray"
                                    Padding="7"
                                    HeightRequest="80"
                                    WidthRequest="300"

AbsoluteLayout.LayoutFlags="PositionProportional"
                                    AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                                    BackgroundColor="LightGray"
                                    HasShadow="True"
                                    CornerRadius="15"
                                    IsVisible="true"
                                    HorizontalOptions="Center">

                            <Frame      BorderColor="LightGray"
                                        Padding="7"
                                        HeightRequest="80"
                                        WidthRequest="310"

AbsoluteLayout.LayoutFlags="PositionProportional"

AbsoluteLayout.LayoutBounds="0.5,0.03,-1,-1"
                                        BackgroundColor="LightGray"
                                        HasShadow="True"
                                        CornerRadius="15"
                                        IsVisible="true"
                                        HorizontalOptions="Center">
                                <Frame.Background>
                                    <LinearGradientBrush EndPoint="0,1">
                                        <GradientStop Color="White"
Offset="0.9" />

                                        <GradientStop Color="LightBlue"
Offset="1.0" />

                                    </LinearGradientBrush>
                                </Frame.Background>

                                <Label      Text="After a Wildfire"
                                            FontSize="26"
                                            HorizontalOptions="Center"
                                            VerticalOptions="Center"
                                            TextColor="#282828"
                                            FontAttributes="Bold">
                                </Label>
                            </Frame>
                        </Frame>

                        <ScrollView>

                            <StackLayout>

                                <Label      Text="Once a wildfire has finished,
    you should only return if it is safe to do so."
                                            TextColor="Black"
                                            FontSize="18"
                                            HorizontalTextAlignment="Center"
                                            IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                            HorizontalOptions="Center"
                                            HeightRequest="1"
                                            WidthRequest="300"
                                            Color="#5b5d68">
                                </BoxView>

                                <Label      Text="Here are some tips to make
    your return as safe and as stress free as possible"
                                            TextColor="Black"
                                            FontSize="19"
                                            FontAttributes="Bold"
                                            HorizontalTextAlignment="Center"
                                            IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                            HorizontalOptions="Center"
                                            HeightRequest="1"
                                            WidthRequest="300"
                                            Color="#5b5d68">
                                </BoxView>

                                <Label      Text="- Listen to the
    professionals."

                                            TextColor="DarkRed"
```

```
                                        FontSize="20"
                                        FontAttributes="Bold"
                                        IsVisible="true">
                                </Label>

                                <Label      Text="Make sure your area has been
cleared by qualified personal before you return."
                                        TextColor="Black"
                                        FontSize="16"
                                        FontAttributes="None"
                                        IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                        HorizontalOptions="Center"
                                        HeightRequest="1"
                                        WidthRequest="300"
                                        Color="#5b5d68">
                                </BoxView>

                                <Label      Text="- Document the damage"
                                        TextColor="DarkRed"
                                        FontSize="20"
                                        FontAttributes="Bold"
                                        IsVisible="true">
                                </Label>

                                <Label      Text="Once you have returned you
should do a thorough inspection and note all the damage dealt. This may help
with insurance claims"
                                        TextColor="Black"
                                        FontSize="16"
                                        FontAttributes="None"
                                        IsVisible="true">
                                </Label>

                                <BoxView    VerticalOptions="Center"
                                        HorizontalOptions="Center"
                                        HeightRequest="1"
                                        WidthRequest="300"
                                        Color="#5b5d68">
                                </BoxView>

                                <Label      Text="- Remain vigilant"
                                        TextColor="DarkRed"
                                        FontSize="20"
                                        FontAttributes="Bold"
                                        IsVisible="true">
                                </Label>
                                <Label      Text="It is important to keep a
close eye for the next few days to ensure nothing has evaded the professional
inspection."
                                            TextColor="Black"
                                            FontSize="16"
                                            FontAttributes="None"
                                            IsVisible="true">
                                </Label>

                            </StackLayout>
                        </ScrollView>
                    </StackLayout>
                </Frame>
            </Frame>
        </AbsoluteLayout>
    </ContentPage.Content>
</ContentPage>
```

Figure 74 - WildlandInfoView.xaml

5.1.5.39. WildlandInfoView.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   WildlabdInfoView
* Purpose:     Backend for Wildland safety info.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire.Views
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class WildlandInfoView : ContentPage
   {
       public WildlandInfoView()
       {
           InitializeComponent();
       }

       // Back Button Event Handler
       private async void BackButton_Clicked(object sender, EventArgs e)
       {
           try
```

```
    {
        await Navigation.PopModalAsync();
    }
    catch(Exception ex)
    {
        ex.Message.ToString();
    }
}

// Before Button Event Handler
private void BeforeFireInfo_Clicked(object sender, EventArgs e)
{
    try
    {
        FrameLabel.IsVisible = false;
        FrameLabel1.IsVisible = false;
        beforeInfo.IsVisible = true;
        BeforeFireInfo.IsVisible = false;
    }
    catch (Exception ex)
    {
        ex.Message.ToString();
    }
}

// Before Remove Event Handler
void BeforeRemovePopupTapped(object sender, EventArgs e)
{
    FrameLabel.IsVisible = true;
    FrameLabel1.IsVisible = true;
    beforeInfo.IsVisible = false;
    BeforeFireInfo.IsVisible = true;
}

// During Button Event Handler
private void DuringFireInfo_Clicked(object sender, EventArgs e)
{
    try
    {
        FrameLabel.IsVisible = false;
        FrameLabel1.IsVisible = false;
        duringInfo.IsVisible = true;
        DuringFireInfo.IsVisible = false;
    }
    catch (Exception ex)
    {
        ex.Message.ToString();
    }
}
```

```
        // During Remove Event Handler
        void DuringRemovePopupTapped(object sender, EventArgs e)
        {
            FrameLabel.IsVisible = true;
            FrameLabel1.IsVisible = true;
            duringInfo.IsVisible = false;
            DuringFireInfo.IsVisible = true;
        }

        // After Button Event Handler
        private void AfterFireInfo_Clicked(object sender, EventArgs e)
        {
            try
            {
                FrameLabel.IsVisible = false;
                FrameLabel1.IsVisible = false;
                afterInfo.IsVisible = true;
                AfterFireInfo.IsVisible = false;
            }
            catch (Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // After Remove Event Handler
        void AfterRemovePopupTapped(object sender, EventArgs e)
        {
            FrameLabel.IsVisible = true;
            FrameLabel1.IsVisible = true;
            afterInfo.IsVisible = false;
            AfterFireInfo.IsVisible = true;
        }
    }
}
```

Figure 75 - WildlandInfoView.xaml.cs

3.1.6.1 App.xaml

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

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

Figure 76 - App.xaml

3.1.6.2 App.xaml.cs

```
/* Author:      Jack McNally
* Page Name:   App
* Purpose:     Main App Backend Page.
*/
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using Wildfire.Services;
using Wildfire.Views;

namespace Wildfire
{
   public partial class App : Application
   {
       IAuth auth;
       public App()
       {
           Device.SetFlags(new string[] {"Brush_Experimental"});
           InitializeComponent();
           auth = DependencyService.Get<IAuth>();
           GoogleMapsApiService.Initialize(Constants.GoogleMapsApiKey);
           MainPage = new MainTabPage();

           if (auth.SignIn())
           {
               MainPage = new SplashScreen();
           }
           else
           {
               MainPage = new SplashScreen();
           }
       }

       protected override void OnStart()
       {
       }

       protected override void OnSleep()
       {
       }

       protected override void OnResume()
```

```
            {
            }
        }
    }
```

Figure 77 - App.xaml.cs

3.1.7 Constants.cs

```
using System;
namespace Wildfire
{
   public static class Constants
   {
       public const string GoogleMapsApiKey = "API_KEY";
   }
}
```

Figure 78 - Constants.cs.

3.1.8.1. MainPage.xaml

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

   <StackLayout>
       <Frame BackgroundColor="#2196F3" Padding="24" CornerRadius="0">
           <Label Text="Welcome to Xamarin.Forms!"
HorizontalTextAlignment="Center" TextColor="White" FontSize="36"/>
       </Frame>
       <Label Text="Start developing now!" FontSize="Title"
Padding="30,10,30,10"/>
       <Label Text="Make changes to your XAML file and save to see your UI
update in the running app with XAML Hot Reload. Give it a try!" FontSize="16"
Padding="30,0,30,0"/>
       <Label FontSize="16" Padding="30,24,30,0">
           <Label.FormattedText>
               <FormattedString>
                   <FormattedString.Spans>
                       <Span Text="Learn more at "/>
                       <Span Text="https://aka.ms/xamarin-quickstart"
FontAttributes="Bold"/>
                   </FormattedString.Spans>
               </FormattedString>
           </Label.FormattedText>
       </Label>
   </StackLayout>

</ContentPage>
```

Figure 79 - MainPage.xaml

3.1.8.2. MainPage.xaml.cs

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

namespace Wildfire
{
   public partial class MainPage : ContentPage
   {
       public MainPage()
       {
           InitializeComponent();
       }
   }
}
```

Figure 80 - MainPage.xaml.cs

3.1.9.1. MainTabPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Author:        Jack McNally
    Page Name:     MainTabPage
    Purpose:       Main navigation bar.
-->
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Wildfire.MainTabPage"
            xmlns:Views="clr-namespace:Wildfire.Views"
            BarBackgroundColor="#ff8c19"

xmlns:android="clr-namespace:Xamarin.Forms.PlatformConfiguration.AndroidSpeci
fic;assembly=Xamarin.Forms.Core"
            android:TabbedPage.ToolbarPlacement="Bottom"
            android:TabbedPage.BarItemColor="Black"
            android:TabbedPage.BarSelectedItemColor="White"
            android:TabbedPage.IsSwipePagingEnabled="False">

 <!--Pages can be added as references or inline-->
   <TabbedPage.Children>

       <Views:InfoView IconImageSource="information.png">
```

```
        </Views:InfoView>

        <Views:MapView IconImageSource="location.png">

        </Views:MapView>

        <Views:SettingsView IconImageSource="settings.png">

        </Views:SettingsView>

    </TabbedPage.Children>
</TabbedPage>
```

Figure 81 - MainTabPag.xaml

3.1.9.2. MainTabPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Wildfire.Views;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Wildfire
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainTabPage : TabbedPage
    {
        IAuth auth;
        public MainTabPage()
        {
            InitializeComponent();
            auth = DependencyService.Get<IAuth>();
            CurrentPage = Children[1];
        }

        private async void SignOut_Clicked(object sender, EventArgs e)
        {
            var signOut = auth.SignOut();

            if (signOut)
            {
                await Navigation.PushModalAsync(new LoginPageView());
            }
        }
    }
}
```

Figure 82 - MainTabPage.xaml.cs

3.2. Wildfire.Android

3.2.1. AuthDroid.cs.

```
/* Author:      Jack McNally
* Page Name:   AuthDroid
* Purpose:     Authentication for Android.
*/
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Xamarin.Forms;
using Firebase.Auth;
using Wildfire.Droid;

[assembly : Dependency(typeof(AuthDroid))]
namespace Wildfire.Droid
{
   public class AuthDroid : IAuth
   {
       // Login
       public async Task<string> LoginWithEmailAndPassword(string email,
string password)
       {
           try
```

```
            {
                var newUser = await
Firebase.Auth.FirebaseAuth.Instance.SignInWithEmailAndPasswordAsync(email,
password);
                var token = newUser.User.Uid;

                return token;
            }
            catch (FirebaseAuthInvalidUserException e)
            {
                e.PrintStackTrace();
                return string.Empty;
            }
            catch (FirebaseAuthInvalidCredentialsException e)
            {
                e.PrintStackTrace();
                return string.Empty;
            };
        }

        // Sign In
        public bool SignIn()
        {
            var user = Firebase.Auth.FirebaseAuth.Instance.CurrentUser;
            return user != null;
        }

        // Sign Out
        public bool SignOut()
        {
            try
            {
                Firebase.Auth.FirebaseAuth.Instance.SignOut();
                return true;
            }
            catch(Exception)
            {
                return false;
            }
        }

        // Sign Up
        public async Task<string> SignUpWithEmailAndPassword(string email,
string password)
        {
            try
            {
                var newUser = await
Firebase.Auth.FirebaseAuth.Instance.CreateUserWithEmailAndPasswordAsync(email
, password);
                var token = newUser.User.Uid;
                return token;
            }
            catch (FirebaseAuthInvalidUserException e)
            {
                e.PrintStackTrace();
                return string.Empty;
            }
            catch (FirebaseAuthInvalidCredentialsException e)
            {
                e.PrintStackTrace();
                return string.Empty;
            }
        }

        // Forgot Password
        public async Task ForgotPassword(string email)
        {
            try
            {
                await
FirebaseAuth.Instance.SendPasswordResetEmailAsync(email);
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }

        // Chnage Password
        public async Task ChangePassword(string newPassword)
        {
            try
            {
                var currentUser = FirebaseAuth.Instance.CurrentUser;

                await currentUser.UpdatePasswordAsync(newPassword);
            }
            catch(Exception ex)
            {
                ex.Message.ToString();
            }
        }
    }
}
```

Figure 83 - AuthDroid.cs

3.2.2. MainActivity.cs

```
/* Author:      Jack McNally
* Page Name:   MainActivity
* Purpose:     Android main activity.
*/
using System;
using Android.App;
using Android.Content.PM;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Firebase;
using Android.Content;

namespace Wildfire.Droid
{
   [Activity(Label = "Wildfire", Icon = "@mipmap/hdpi", Theme =
"@style/MainTheme", MainLauncher = true, ConfigurationChanges =
ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode |
ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize )]
   public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
   {
       protected override void OnCreate(Bundle savedInstanceState)
       {
           TabLayoutResource = Resource.Layout.Tabbar;
           ToolbarResource = Resource.Layout.Toolbar;

           base.OnCreate(savedInstanceState);

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

Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);

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

       protected override void OnNewIntent(Intent intent)
       {
           base.OnNewIntent(intent);
       }
   }
}
```

Figure 84 - MainActivity.cs

3.2.3. NotificationHelper.cs

```
/* Author:      Jack McNally
 * Page Name:   NotificationHelper
 * Purpose:     Notification Builder.
 */
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Wildfire.Services;
using Xamarin.Forms;
using Firebase.Auth;
using Wildfire.Droid;
using Android.Media;
using Android.Support.V4.App;


[assembly: Dependency(typeof(NotificationHelper))]
namespace Wildfire.Droid
{
   class NotificationHelper : INotification
   {
       private Context mContext;
       //private NotificationManager mNotificationManager;
       private NotificationCompat.Builder mBuilder;
```

```
        public static String NOTIFICATION_CHANNEL_ID = "10023";

        public NotificationHelper()
        {
            mContext = global::Android.App.Application.Context;
        }

        [Obsolete]
        public void CreateNotification(String title, String message)
        {
            try
            {
                var intent = new Intent(this.mContext, typeof(MainActivity));
                intent.AddFlags(ActivityFlags.SingleTop);
                intent.AddFlags(ActivityFlags.NewTask);

                var pendingIntent = PendingIntent.GetActivity(this.mContext,
0, intent, PendingIntentFlags.OneShot);

                mBuilder = new NotificationCompat.Builder(mContext);

mBuilder.SetSmallIcon(Resource.Drawable.common_google_signin_btn_icon_dark_no
rmal);
                mBuilder.SetContentTitle(title)
                    .SetAutoCancel(true)
                    .SetContentTitle(title)
                    .SetContentText(message)
                    .SetChannelId(NOTIFICATION_CHANNEL_ID)
                    .SetPriority((int)NotificationPriority.High)
                    .SetVibrate(new long[0])
                    .SetVisibility((int)NotificationVisibility.Public)

.SetSmallIcon(Resource.Drawable.common_google_signin_btn_icon_dark_normal)
                    .SetContentIntent(pendingIntent);

                NotificationManager notificationManager =
mContext.GetSystemService(Context.NotificationService) as
NotificationManager;
                if (global::Android.OS.Build.VERSION.SdkInt >=
global::Android.OS.BuildVersionCodes.O){
                    NotificationImportance importance =
global::Android.App.NotificationImportance.High;

                    NotificationChannel notificationChannel = new
NotificationChannel(NOTIFICATION_CHANNEL_ID, title, importance);
                    notificationChannel.EnableLights(true);
                    notificationChannel.EnableVibration(true);
                    notificationChannel.SetShowBadge(true);
                    notificationChannel.Importance =
NotificationImportance.High;
                   notificationChannel.SetVibrationPattern(new long[] { 100,
200, 300, 400, 500, 400, 300, 200, 400 });

                   if (notificationManager != null)
                   {
                       mBuilder.SetChannelId(NOTIFICATION_CHANNEL_ID);

notificationManager.CreateNotificationChannel(notificationChannel);
                   }
               }
               notificationManager.Notify(0, mBuilder.Build());
           }
           catch(Exception ex)
           {
               ex.ToString();
           }
       }
   }
}
```

Figure 84 - MainActivity.cs

4. Declaration

- I declare that all material in this submission, e.g. thesis/essay/project/assignment, is entirely my own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams, or other material; including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in the preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Student Name : Jack McNally

Student Number : C00228768

Student Signature : [illegible]

Date: 30|04|21