A-Level Computer Science Project

Missing Persons

Ben Gavin

Contents

Analysis

Problem Identification

Currently the police are using the last seen location of a missing person to then draw circles around that point to work out where that person could be now, but the problem is it can be inaccurate as it is down to a policeman or woman to use their judgement of how far the missing person could have gotten. Circles are drawn on a map to show the possible distance covered to give them and search parties boundaries so they don't spend time looking in the wrong place, however this can also be inaccurate and can waste possible searching time, searching in places where the missing person couldn't actually have reached due to there being a steep hill in the way.

Police also have to organise search parties to help come and look for the missing person, this is often done manually by phone or email which can take a lot of time, and then responses have to be read and maybe even responded to.

Stakeholders

The main clients for this are the police. It will be designed, aimed at and around them to make it easy for them to workout boundaries for a missing person based on the amount of time they have been missing and also organising search parties. I have one stakeholder for this part of the project, James, who is a police officer for the Hampshire Police.

Another client are the people who take part in searches or organise search parties. I have gathered a few stakeholders for this part of the project who said they would be willing to participate in search parties. These people are Mr Mapstone, Will and Sava.

Why is this suited to a computational solution?

The problem is suited to a computational solution because it has the ability to drastically improve accuracy and makes it less prone to human error. The solution will be an algorithm that will make a polygon (isochrone) of were that missing person could have got to within a set amount of time, it will also allow people (such as the police) to start a search and have the ability to share it with people so they can help search.

Problem recognition

The main problem is the search area that is made by a human because it can be inaccurate as it is based on their inquisition. Another problem related to a missing persons case is organisation and communication, so people are often individually called or emailed (manually) which is not efficient as the police (and search parties) need to be using that time well as a missing person could be travelling further and further away.

Problem decomposition

The problem can be broken down into smaller steps in order to make it easier to approach:

1. Take input parameters (such as last seen location, speed, start time, etc.) from the user
2. Run these parameters through an algorithm that will take into account terrain data from around the start location
3. Then plot the search area created by an algorithm on a map their location on it too for reference

Interview

Interview Questions

To start off the interview, I will explain the concept behind the app and the features of both the coordinator and searcher app.

1. Is this software something that you could see yourself using and if not, why not?
2. Are there any other features that you would like to see in the application?
3. Do you think this software will benefit the police and also search parties?
4. What are your thoughts on this just being a mobile application?
5. Are there any other comments you would like to add?

Interviews

Police officer:

James:

1. **Is this software something that you could see yourself using and if not, why not?**
   Yes it is definitely something I could see myself using.
2. **Are there any other features that you would like to see in the application?**
   I think an implementation of something such as what3words would be a good idea as well as the application showing nearby hospitals if the person is injured, for example.
3. **Do you think this software will benefit the police and also search parties?**
   I think it has the potential to benefit the police, especially in terms of being able to share the isochrone with search parties.
4. **What are your thoughts on this just being a mobile application?**
   I personally think it is fine as a mobile application as it means we can all have it on our phones and carry it with us, however I think it would be useful if this information was available on a computer as well.
5. **Are there any other comments you would like to add?**
   Yes, I think that self-expanding search areas would be a good feature to add so that we don't have to adjust the time ourselves.

Searchers:

Mr Mapstone:

1. **Is this software something that you could see yourself using and if not, why not?** Yes, based upon the current problem this seems a much more intuitive way of mapping out a perimeter, especially due to the time sensitive nature of a search.

2. **Are there any other features that you would like to see in the application?** Perhaps in future using machine learning - based on previous missing person to highlight more likely areas or routes to search, perhaps giving percentage likelihoods.
3. **Do you think this software will benefit the police and also search parties?** From the current solution to the problem, yes. Easier and quicker.
4. **What are your thoughts on this just being a mobile application?** Makes sense as a solution needs to be portable due to the nature of the problem situation, the officers being out and about.
5. **Are there any other comments you would like to add? n/a**

Will:

1. **Is this software something that you could see yourself using and if not, why not?**
   Yes probably, as it means I can just participate in a search and it makes it easier for the police to update me on search areas.
2. **Are there any other features that you would like to see in the application?**
   The ability to press a location and be able to start a search like that as well as entering an address. Also a filter, such as searches near me so that ones very far away don't come up for me as I won't be able to participate.
3. **Do you think this software will benefit the police and also search parties?**
   Probably if you can get them to incorporate to routine.
4. **What are your thoughts on this just being a mobile application?**
5. **Are there any other comments you would like to add?**
   The algorithm should run on the phone, so that when there is no internet connection or signal it will still be able to load in a search area.
   Another idea I think should be implemented is a self expanding search area based on the starting time.

Sava:

1. **Is this software something that you could see yourself using and if not, why not?**
   If it were my job to search for people, then I can see myself using this software.
2. **Are there any other features that you would like to see in the application?**
   The ability to add additional isochrones on top of the current one from a user selected point on the map and to be able to determine walking distance from your current location to elsewhere.
3. **Do you think this software will benefit the police and also search parties?**
   I think it would be very beneficial to police and/or organisations.
4. **What are your thoughts on this just being a mobile application?**
   I think having a mobile application is ideal, however I would like to see there being a desktop version for use within a command information centre such as a police operation centre.
5. **Are there any other comments you would like to add?**
   Sounds interesting and should be a good project.

Stakeholder Requirements

Requirement	Explanation
Simple design	The design must be clean and easy to navigate around
Offline functionality	The application must be able to do most of its processing offline so when there is no data connection a search area can still be plotted on a map
Self-expanding search area	The isochrone should expand by using the difference between the starting time and current time as the total time
Adding markers using finger	The map must listen to when a user taps on the screen and plot a marker there as well as being able to put a starting marker by inputting an address into the search box

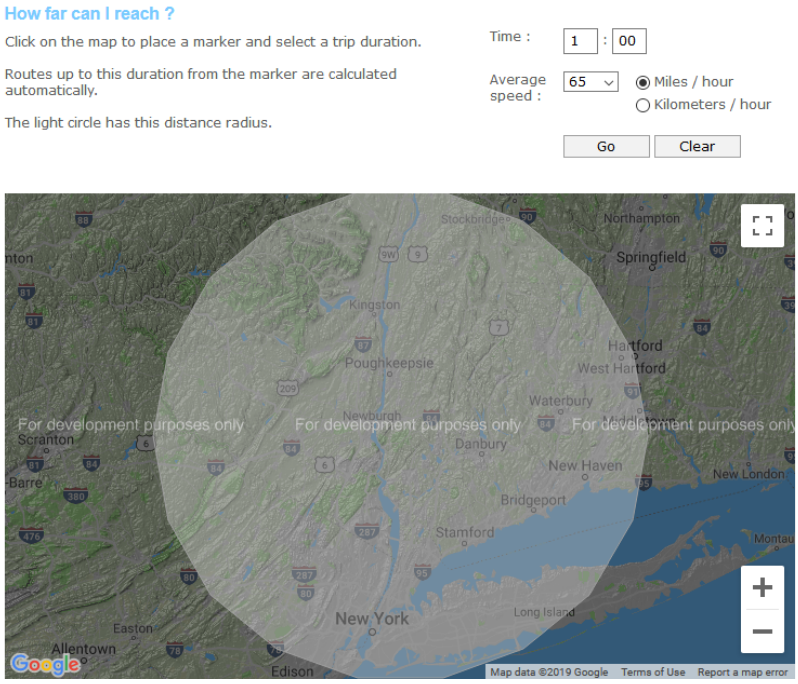
Research

Existing similar solution

Isochrones.com

Overview:

This site has a simple embedded google map and two parameters, time and average speed. Whilst this site does make an isochrone, it doesn't have the best accuracy as it makes a polygon circle, which is fine, however if this were to be used for missing persons it isn't really going to narrow down the places that someone could've gotten to, as it looks like elevation data isn't be factored in. To start making the isochrone, the user must click on the map where they want to start, which is intuitive but not applicable for me as this can make it harder to pick a location whilst on a mobile device.



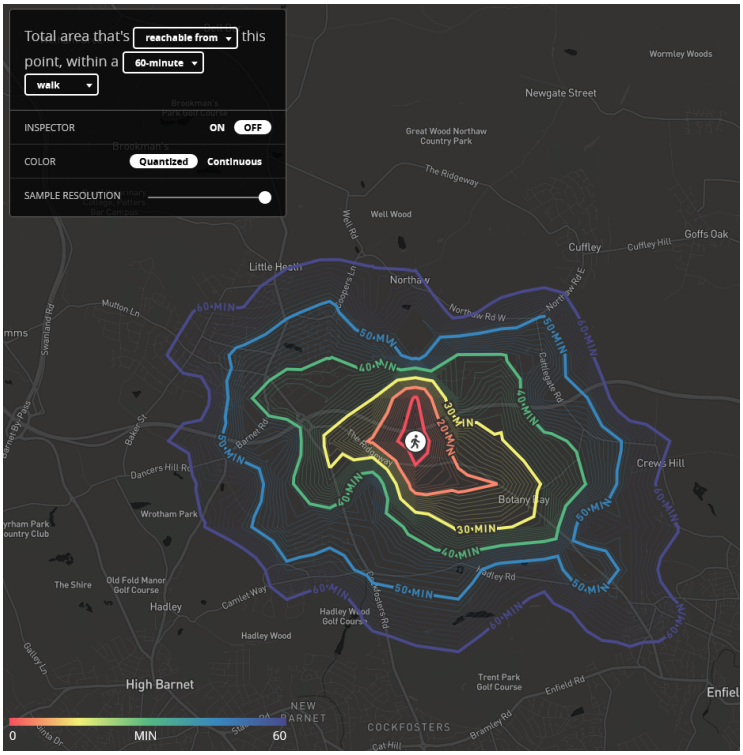
What can I incorporate?

My solution will need to have input from the user for the starting point, however I don't think it should be through tapping the screen where you want to start even though this is intuitive, but on a (small) mobile device this could be inaccurate, so I think inputting an address through a textbox is much easier and avoids all inaccuracies. As for parameters, I think incorporating the average speed is a must have as if the missing person were to have mobility problems (broken legs etc) then they won't be able to get as far so this is definitely an import factor. However I think inputting the time to start could be a bit of a nuisance as the coordinator might want to extend the time with maybe a slider instead of having to keep recreating an isochrone with a different time.

Mapbox

Overview:

Mapbox are a service that I am planning to use to load in the map into the app. They also provide an isochrone API with an example at https://labs.mapbox.com/bites/00156/. The example is more intuitive to use than isochrones.com and factors in terrain making it more suited to a missing persons case. There are more parameters for this site which include time, mode of transport, different ways of viewing the data and the ability to change the sample resolution. This tool is easy to use and read which means it is user friendly.



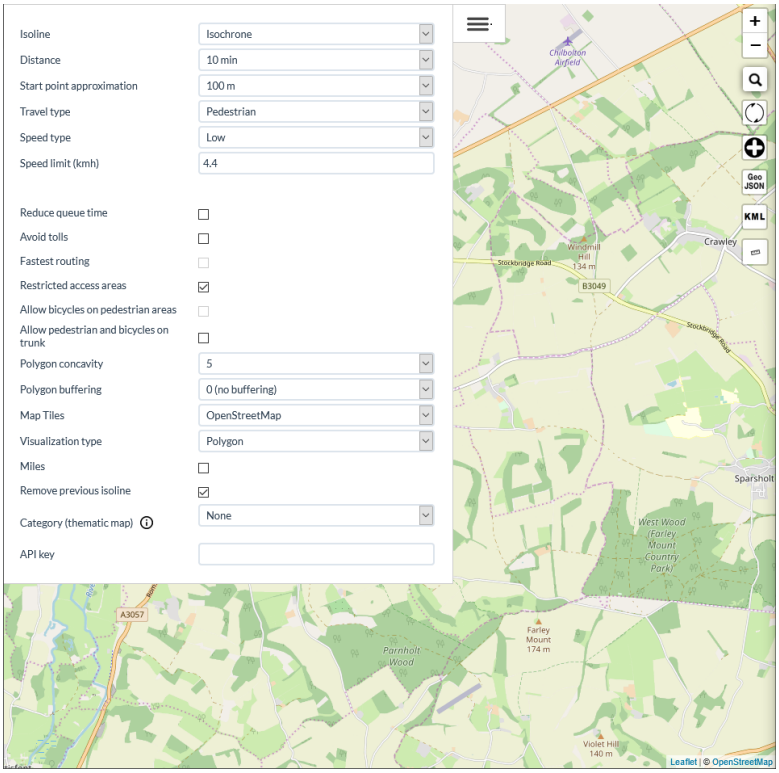
What can I incorporate?

My solution could incorporate some of these extra parameters provided by mapbox, such as mode of transport as the missing person could have used a bike or car. I also think the ability to adjust the sample resolution is a good idea, but not necessarily worth adding because if everyone were to have slightly different sample resolutions, it would mean they would all have slightly different search boundaries. The limitation of this site is the time that you can input, it is capped at 60 minutes, which is not very helpful for police as a person can go missing for many hours so my algorithm needs to be able to support longer times (which may affect the efficiency of the algorithm).

Iso4App

Overview:

This isochrone generator has a lot more input parameters available, which makes it a little less user friendly and less intuitive than the mapbox one. You are able to input speeds and they also have presets if you don't know what the exact speed would be. There are also other boolean parameters such as avoid tolls and allow bicycles on pedestrian areas which are helpful as they can change the isochrone generated, possibly quite drastically depending on the surrounding area.



What can I incorporate?

My solution will use speed as a factor to generate the isochrones but it I think having the presets is a good idea, especially in the case of a missing person as the police will not know the exact speed they are going at but may have an idea in terms of slow, medium or fast.

Proposed Solution

Features of my proposed solution

My solution will be in the form of a mobile application. Upon starting the app, the user will be greeted with two tabs, the being a map with a search box at the top. The search box will allow the user to enter an address which will turn the address into coordinates in order to plot it on a map. Once a point has been plotted on a map, the user is able to create a search from here, inputting data such as speed and other factors. On the second tab will be a list of current searches started by the user (along with other searches shared by other users). The user should be able to click on a search to take them back to the map and show the isochrone.

Limitations of my proposed solution

One limitation is that this is only a mobile app so will not be able to be used on a computer, which for some people might be a bad thing as they might think phones are inaccurate and it is hard to do stuff on them.

Another limitation of this is an internet connection. The algorithms will all be run on the phone so that means any stored information (in a database) will be fine as the algorithm itself will not require an internet connection, however in order to start and share any new

searches an internet connection would be required to store it on a server (so other users can see it).

Requirements

Hardware requirements

An Android phone with internet access - The application needs internet access in order to turn the address into coordinates and also contact a central server where searches that need shared are stored.

Phone recommended hardware requirements:

- CPU - Dual-core 1.1Ghz
- RAM - 1GB
- GPS (with higher accuracy using networks is optional)
- Touch screen (size does not matter)

The phone has these recommended requirements because it needs to be able to load and then run an interactive map. The CPU speed only plays a part in how fast it loads as well as the RAM, although the app is fairly memory efficient as there are not many global variables. The GPS is only recommended, but not required. It is used for determining the current location of the user and plotting it on the map, although if disabled in settings or not available, the location will not be plotted. A touch screen is required in order to interact with the map itself, like being able to tap in order to app a marker, or zooming in on the map by pinching with two fingers.

Software requirements

Android - The user must have android at a version of at least 4.0.3 in order to run the application.

4.0.3 is the minimum requirement because the application will be compiled with API 15 (Ice Cream Sandwich), this is due to the libraries that will be used (e.g. map and database). Any lower and the app will no longer run, although being above 4.0.3 means that in the future, if more libraries were added the user is more likely to be able to receive these updates.

Success Criteria

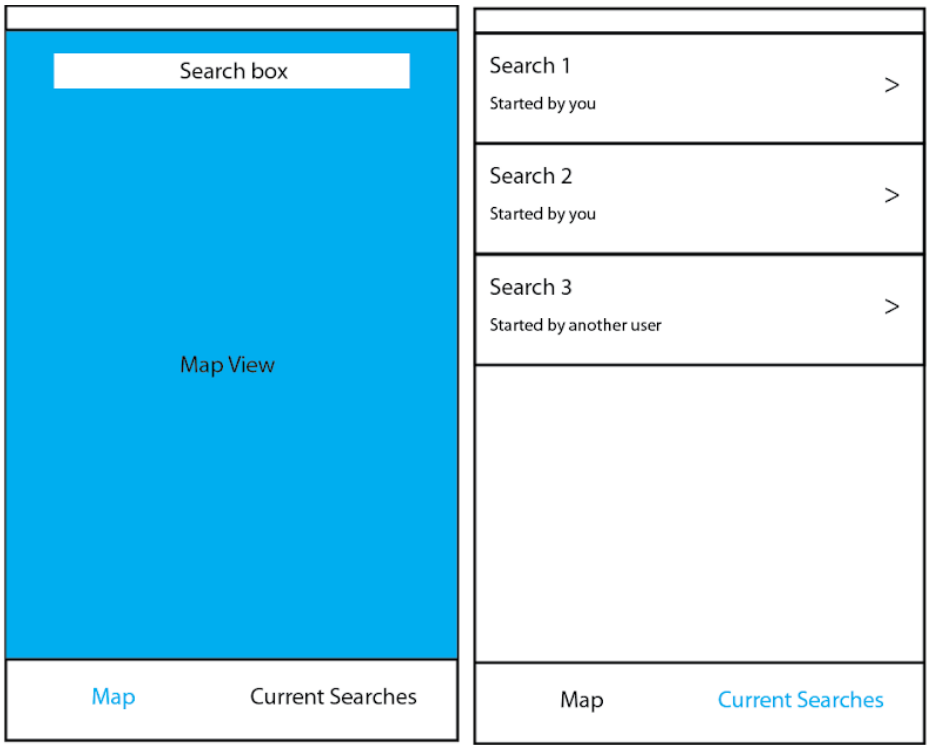
No.	Criteria	Evidence
1	Interactive map	It must have an interactive map because then the user is able to see the search boundaries.
2	User location on map	The user should be able to see their live location so that

		they can see where they are inside or outside of the search boundary.
3	Use an address to plot a marker	The user should be able to type an address and the application will turn this into coordinate. This is so the coordinates can be used to plot a marker on the map which will be used as a starting point for a search.
4	Tap the map to plot a marker	The user should also be able to tap a position on the map and add a marker there. This can also be used to start a search.
5	Simple design	The application should have a simple design and clear legible text so the user can work out what needs to be done without loads of instructions.
6	Search parameter input	The user should be able to input a list of parameters on a screen which will be used by the algorithm. This is so the algorithm can be adjusted for each missing person as they will have different walking speeds etc
7	Isochrone algorithm	The algorithm should be able to take input parameters and return a dataset with the coordinates for a polygon to be made.
8	Fetching elevation data	The algorithm will need to use elevation data to work out how far the person can get by using openly available data.
9	Display search boundary	The app should clearly display the estimated search boundary (polygon) on the map which indicates the

		search boundaries.
10	Current searches	The application should list the current searches started by them and others. This is so other people can help with searches and also so multiple searches can be stored and loaded later when needed.
11	Search Insight	This should allow the user to see more detailed information than just listed on the current searches screen, such as name and starting position etc.
12	Saving searches to an online database	The searches should be saved to an online database so that they are backed up, it also allows for other users to see the search if they want to join in and help to look.
13	Deleting searches	The owner of the search should be able to remove the search from the online database. This could be used, for example, if a search has finished and the person has been found.
14	Contacting the search owner	The user should be able to have some form of communication button inside the app that will allow them to send pictures of any evidence or send them any findings etc.

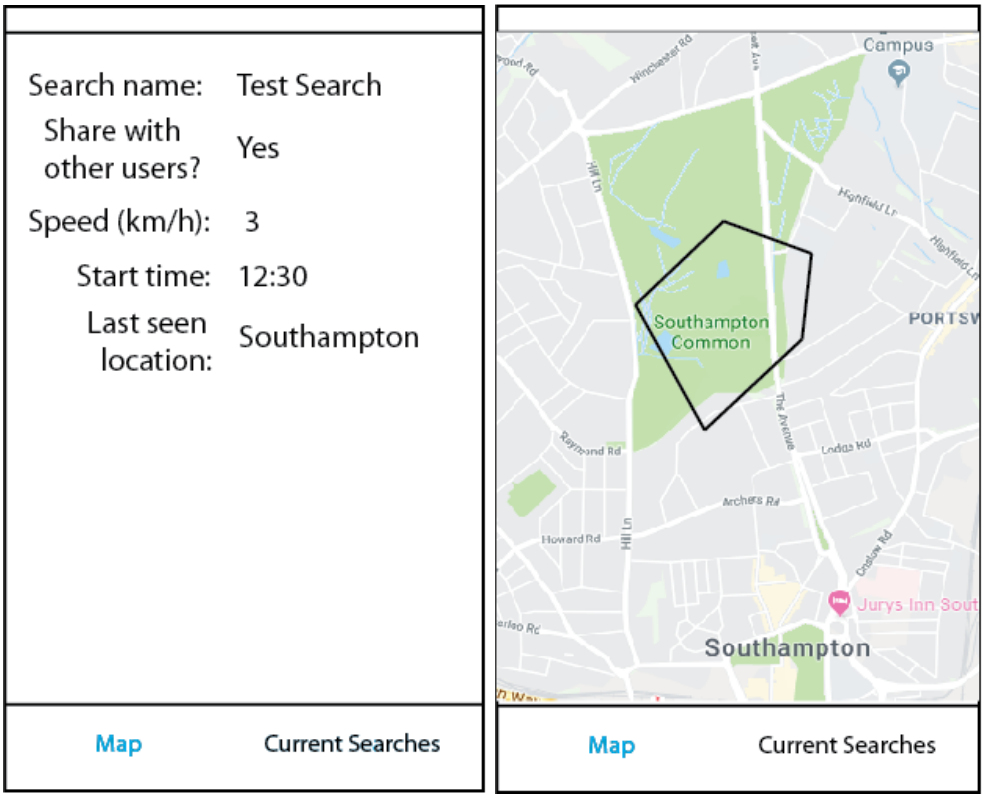
Design

App UI Design



This is the main screen of the app and this is what the user will see when they start it. A menu wasn't necessary for my solution but I thought a navigation bar at the bottom would suit it better. There are two tabs, "Map" and "Current Searches", the map is the picture on the left and will have a map view with the users location plotted on the map at the start, it will also have a search box where the user can input an address and a marker will be plotted on the map. This links to the stakeholder requirements of it being a simple design and it links to the success criteria of having a map with the users location on it.

The second tab (shown on the right) is the current searches, which will list both searches started by you, and also one's shared by other people who are trying to get people to help with the search.



On the left is the new search screen which has a fairly simple layout and just gets some parameters from the user in order for the algorithm to work. The search name field is a text box that means the user can easily identify the search on the search list. The share with others users is a boolean input and this asks the user if they want their search to show up on other people's current searches tab or not. The speed can be an int or a float and is the main factor in producing the isochrone. The start time allows the user to input a custom start time, although this will automatically be filled in with the current time. The last seen location will be set to the location from the main map screen with the search box.

The screen on the right shows an example of what the user will see once the isochrone algorithm has finished running. It will show the map and a plotted search boundary of where the missing person could be.

Stakeholder Input

Now that I have my initial UI designs I can contact my stakeholders and ask them their thoughts on the app and tell me if any improvements need to be made or if something needs to be added/removed.

James: *Initial screens look promising. I guess the ability to mark where has been covered is important so that other teams do not repeat covered zones. Levels of search is also important, visual search through to fingertip can all be important. The distance travelled and perhaps even bus routes, car parks are important but will come up on the maps. Are you going to have the ability to put a photo of the missing person on the app as if officers are using it they can show the image to anyone they see in the area and ask if they have seen the missing person?*

Mr Mapstone: *Very good and clear UI design. Maybe improve some of the text alignment on the start search screen.*
Sava: *Overall, it's a good UI design and I would be happy to use the solution if this was the final implemented design. However, I would like to see more differentiation between the two buttons at the bottom of the screen by including a vertical line between the two. In addition, I would like to have a more rounded search box. I would prefer to see the inputs on the third screen to be more obvious and intuitive, maybe by having a border around the input area; I'd also like to see the headings be on a single line and not multiline.*
Will: *It is definitely a very clear design however it is a little boring. I think it needs to be clear that the values can be edited by displaying them in a textbox.*

Algorithms

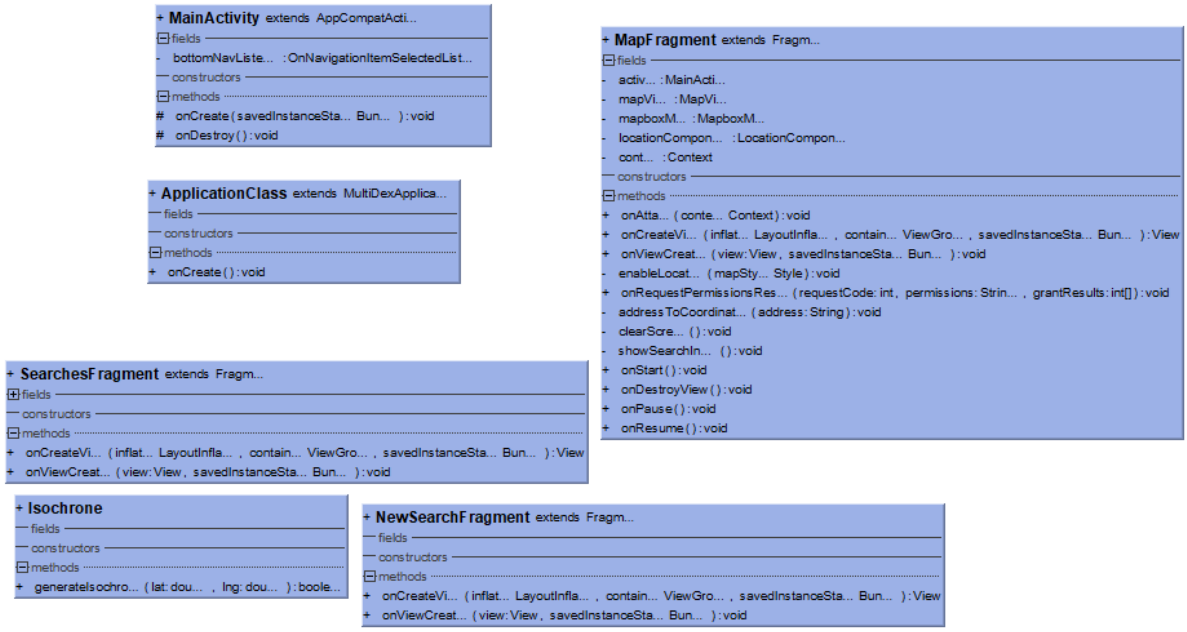
It is important that the application is clean, simple and easy to use however the core solution of this problem is the algorithms used to calculate the search boundary.

The app needs to be able to take inputs (such as speed and start location) and output a search boundary.

Some input validation will be handled by android as when making TextEdits, a type can be specified such as number or text and that will automatically bring up the correct keyboard layout and only let users type numbers for example.

```
<EditText
    android:id="@+id/search_name_input"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="0.50"
    android:inputType="text"
    android:ems="10" />
```

Class Diagram:



Classes:

Class	**Explanation**
ApplicationClass	Run on startup, referenced in Manifest. It is needed in order to initialise and create an instance of the map using an API token. Does not inflate any layout files or invoke any other classes.
MainActivity	Extends AppCompatActivity. Creates the main activity for the app (See Main Activity in layout files) and sets up a listener for the bottom navigation bar. On start, it will also invoke the MapFragment class.
MapFragment	Extends Fragment. Handles android fragment lifecycle methods. Inflates the map layout file (See Map Fragment in layout files). It also plots the user's location on the map but checks for permissions first. This class will also be responsible for making API calls to turn the address into coordinates and plot them.
SearchesFragment	Extends Fragment. Handles android fragment lifecycle methods. Inflates the searches fragment layout file (See Searches Fragment in layout files). This class will also contact a remote NoSQL database and a local SQLite database.

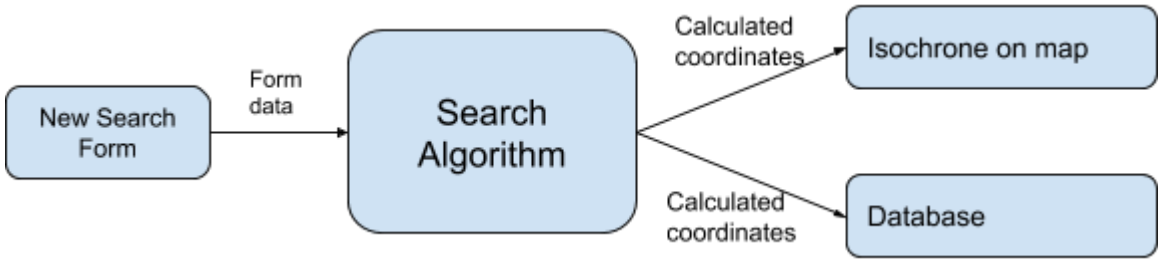
NewSearchFragment	Extends Fragment. Handles android fragment lifecycle methods. Inflates the new search fragment layout file (See New Search Fragment in layout files). It will also create a new instance of the Isochrone class.
Isochrone	Does not extend any class or inflate any layouts. This class will run all the algorithms responsible for making the end isochrone. It will fetch the elevations of coordinates and calculate the end coordinates for the points of the polygon.

Layout files:

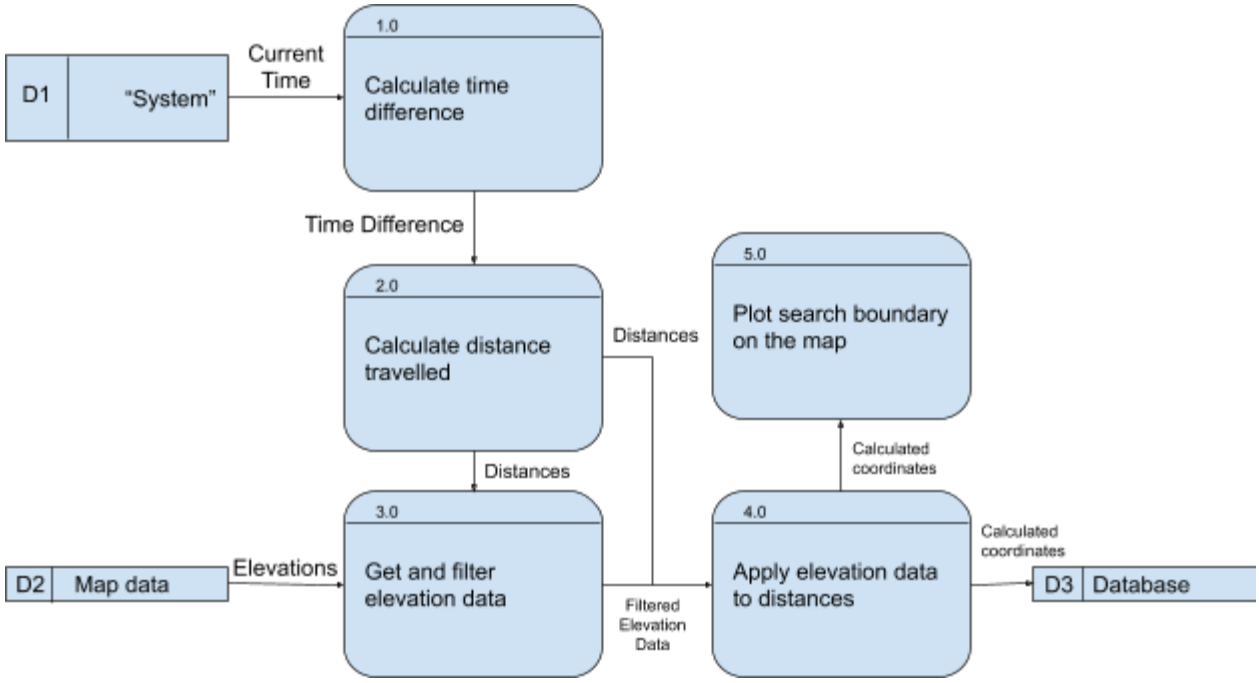
Layout	**Explanation**
Main Activity	The main activity is simple and will just contain a FrameLayout and a BottomNavigationView which will allow the user to navigate between the map view and a list of searches.
Map Fragment	The map fragment will consist of a RelativeLayout with an EditText child for the user to input an address in order to start a search.
Searches Fragment	The searches fragment will have a LinearLayout and have a ListView child which will be used to add the searches from a database.
New Search Fragment	The new search fragment will have a LinearLayout with more LinearLayouts as children to display a series of TextViews and TextEdits which will be used to get inputs.

I have split the classes up like this because when developing an Android app, it is good practice to have each individual activity/fragment (screens) have their own class, this makes it easier to read/follow and debug. Making each screen with have its own class means that I am able to inherit different classes to make the screen act differently (e.g. extends Fragment vs extends Activity).

Here is a level 0 data flow diagram:



Here is a level 1 data flow diagram:



Step 1.0 will calculate the time difference between the time now and the time entered by the user on the “New Search” screen. I plan to get the current time by using System which holds the current time in unix.

Step 2.0 will use this time difference, along with the entered speed to calculate the distance travelled from the last seen location to point X by using “distance = speed * time”. This step will also need to convert this distance before going on to the next step.

Step 3.0 will use a REST API to get and filter elevation data, which will require parsing of a JSON object returned from the server. Once parsed, this will then need to be iterated over to get only the data needed for step 4.0.

Step 4.0 is where the elevation data gets merged with the calculated distance travelled to create points which will equate to a visual search boundary.

Step 5.0 will take the coordinates calculated by 4.0 and plot them on a map using the map SDK.

I have broken down the problem like this because it is now in more manageable chunks, which makes it easier to think about whilst writing the code for the sub-procedure. For example, two big chunks are fetching the elevation data and then applying it. It is best that these were two separate procedures as it means I am less likely to make mistakes and it makes the code easier to read as well.

Pseudocode:

Step 1.0:

```
private long timeDifference(long startTime) {
      startTime = startTime/1000L; //Turns ms into s
      long unixTime = System.currentTimeMillis() / 1000L; //Fetches system date in s
      long difference = unixTime - startTime;
      return difference;
}
```

Step 2.0:

```
private double getDistanceTravelled(double timeDifference, double speed) {
      double displacement = timeDifference * speed;
      return displacement;
}
```

Step 3.0:

```
private float fetchElevation(float[] coord) {
            String url = "api url";
            HTTPClient = new HTTPClient();
            String response = HTTPClient.request(url);
            JSONObject jsonObject = JSON.parse(response);
            float elevation = jsonObject.elevation;
            return elevation;
}
```

These variables are needed for step 4.0 and 5.0

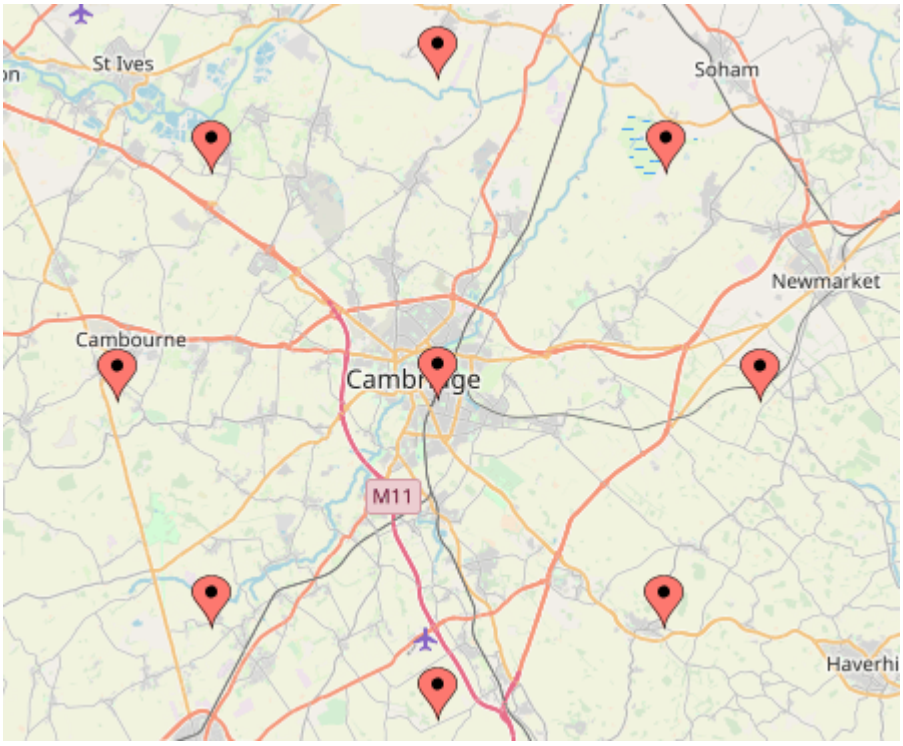
```
private List<List<Point>> POINTS = new ArrayList<>();
private List<Point> OUTER_POINTS = new ArrayList<>();
```

Step 4.0

Before elevations can be applied to distance data, distances from a variety of bearings must be calculated in order to get the outer bounds of the search area. For now, there will be 8 points calculated.

```
private double toRadians(double degrees) {
    double radians = degrees * ((Math.PI)/180);
    return radians;
}

private double toDegrees(double radians) {
    double factor = 180 / Math.PI;
    return radians * factor;
}

private ArrayList<Double> getMaxCoordinates(double[] startingPair, double distance) {
    double lat = toRadians(startingPair[0]);
    double lon = toRadians(startingPair[1]);
    double distanceRatio = distance/6371.01; //Divide the distance (km) given by the radius of the Earth (km) to get a ratio
    ArrayList<Double> endMaxCoordinates = new ArrayList<Double>();
    for(int i = 0; i < 360; i+= 45) { // < 360 as 360 should not be included as this is the same as 0
        double bearing = toRadians(i);
        double endLat = Math.asin(Math.sin(lat)*Math.cos(distanceRatio) + Math.cos(lat)*Math.sin(distanceRatio)*Math.cos(bearing));
        endMaxCoordinates.add(toDegrees(endLat));
        double endLon = lon + Math.atan2(Math.sin(bearing)*Math.sin(distanceRatio)*Math.cos(lat), Math.cos(distanceRatio)-Math.sin(lat)*Math.sin(endLat));
        endMaxCoordinates.add(toDegrees(endLon));
    }
    return endMaxCoordinates;
}
```

Upon testing this method by calling “double[] arr = {52.20472, 0.14056}; <Isochrone>.getMaxCoordinates(arr, 15)” it came back with the results below (points were manually plotted at this point, but will be automated later)



Once these points have been fetched I can then apply elevation data that was fetched earlier. I plan to use the rule for every 10m in altitude change add 1 minute of travel time on in order to adjust the isochrone.

```
public ArrayList<Double> applyElevationsToPoints(ArrayList<Double> pointCoordinates, ArrayList<Integer>
elevations, double speed) {
    int j = 0;
    int k = 1;
    int elevationAtStartPoint = elevations.get(elevations.size()-1);
    ArrayList<Double> endCoordinates = new ArrayList<>();
    for(int i = 0; i < 8; i++) {
        int bearing = 0;
        double lat = pointCoordinates.get(j);
        double lon = pointCoordinates.get(k);
        int elevationAtPoint = elevations.get(i);
        int heightDifference = elevationAtStartPoint - elevationAtPoint;
        int minutesToAdd = heightDifference / 10;
        double distanceTravelled = getDistanceTravelled(minutesToAdd * 60, speed);
        ArrayList<Double> retractionCoordinates = retractDistance(bearing, lat, lon, distanceTravelled);
        double endLat = retractionCoordinates.get(0);
        endCoordinates.add(endLat);
        double endLon = retractionCoordinates.get(1);
        endCoordinates.add(endLon);
        j = j + 2;
        k = k + 2;
        bearing = bearing + 45;
    }
    return endCoordinates;
}
```

Where retractDistance() is quite similar to getMaxCoordinates but inverts the bearing first, which is crucial as then it will calculate the distance in order to move that point back by.

```
private ArrayList<Double> retractDistance(int bearing, double lat, double lon, double distance) {
    lat = toRadians(lat);
    lon = toRadians(lon);
    double oppositeBearing = toRadians(360 - bearing); // Invert bearing/direction
    double distanceRatio = distance/6371.01;
    ArrayList<Double> endCoordinates = new ArrayList<>();
    double endLat = Math.asin(Math.sin(lat)*Math.cos(distanceRatio) +
Math.cos(lat)*Math.sin(distanceRatio)*Math.cos(oppositeBearing));
    endCoordinates.add(toDegrees(endLat));
    double endLon = lon + Math.atan2(Math.sin(oppositeBearing)*Math.sin(distanceRatio)*Math.cos(lat),
Math.cos(distanceRatio)-Math.sin(lat)*Math.sin(endLat));
    endCoordinates.add(toDegrees(endLon));
    return endCoordinates;
}
```

Key Variables

Variable	Data Type	Explanation
speed	double	Used for isochrone algorithm, to calculate distances travelled. This value is taken from the user as an input.
startTime	double	This value is supplied by the user but will need to be converted into seconds and a standard for this is epochs (unix).
lastSeenLocation	(Double) array containing latitude and longitude	This value will be inputted by the user and then turned into coordinates using an API.
timeDifference	double	Value calculated by computer using start time and time now. Value is in epoch seconds. This value will be used as a parameter for step 1.0
elevationData	JSONObject	This is an object returned from an API call that will contain all of the elevation data
endMaxCoordinates	ArrayList<Double>	An array holding the maximum possible distance that could be walked within x time (no elevation data applied yet). They are at 45 degree increments.

points	List<List<Point>>	A list containing outerPoints
outerPoints	List<Point>	A nested list containing points of polygon (search area)
map	Object (MapView)	An object from the Mapbox SDK that will produce an interactive map
location	Object (LocationEngine)	This will hold the live user location which will get updated by the mapbox SDK periodically.

Validation

In order to ensure the application doesn't crash because of the data input by a user, inputs will need to be validated.

Buttons

Buttons are managed by Android. A button has no functionality without any listeners for it, so on start, I need to use the "<Button>.setOnClickListener" for all of the buttons and link them to another method. Buttons do not contain any data so there is no risk of them passing in any wrong inputs into any methods.

Text boxes

Text boxes will need to be validated because they could contain characters that might crash the application. I will use regular expressions in order to filter out the user input. A regular expression is a search pattern which can be used for finding and replacing things within strings. Some of the validation can be handled by Android because you can set the input type to only be a number for example, and it won't allow the user to type letters in the box. If there is an invalid input, then a dialog will be presented letting the user know so they can change it if there is a possible mistake. However, for the address search box at the top of the map screen I will strip the address to remove any apostrophes or escape characters as it might cause the API to throw an error.

Dropdowns

Dropdown boxes are used for inputting whether or not to share the search with other people. This doesn't need input validation as it is an array string so has preset values.

Testing Methods

Whilst developing the application, it is important to check that each method works as it is added. By testing each method as it is made, it will reduce the amount of debugging at the end of the project and it will be easier to see where the errors come from.

Testing will include inputting data into text boxes to see if the output is expected or not and if any exceptions are thrown. Abnormal inputs such as emojis (unicode) should be included in the text box testing.

All test data and outputs will be recorded to compare against expected outputs and if anything is not working as it should then it will be fixed in the next version in the iterative development section.

Testing checklist

The main thing that needs to be tested are buttons and text boxes.

Type of data	**Data to test**	**Expected outcome**	**Actual outcome**
Valid	test search	Continue as normal	
Valid	James' search	Continue as normal	
Valid extreme	@#3482<<//;search|	Continue as normal	
Erroneous	😬ŋИΘ	Continue as normal	

Whilst I will carry out testing, everything should continue as normal for text input. This is because in java, if something is explicitly a string, it will get stored as a string so emojis or other unicode characters will get escaped by Java. Casting can also enforce this, although is likely unnecessary in most cases as unicode characters will be escaped. The last row is marked as erroneous as it should continue as normal and not cause anything to break or crash, however the user is unlikely to actually enter data like this.

Buttons will be tested just by pressing them and making sure that they do the right thing without fail.

Development and Testing

As the platform I am developing for is Android, the project will likely contain many files for each class. The GUI is all managed by the android library and will likely be referenced in all of the class files.

Iteration 1 - Basic Search Area on Map and Current Searches

For this iteration I will have the layout made (navigation bar) and have 2 fragments, one that has the map and one with the current searches (this iteration will include fetching the current searches from an online database and displaying them for the user). The user should be able to see a basic isochrone on a map, tap the map to plot a marker, use a search bar to input an address, input data on the start search screen, see their live location on the map and see current searches at the end of this iteration.

Code

The final code for this iteration can be found in Final Code - Iteration 1 - Basic Search Area on Map and Current Searches.

First I made an application class to initialise the map throughout the whole application.

```
public class ApplicationClass extends MultiDexApplication {

        //Referenced in manifest
        //Initialises map in the onCreate method
        @Override
        public void onCreate() {
        super.onCreate();
        String token = getString(R.string.mapbox_mapsdk_token);
        Mapbox.getInstance(getApplicationContext(), token);
        }

}
```

Next I created my MainActivity class which is an activity to hold all of the fragments in it. It also listens for clicks on the bottom navigation bar.

```
private BottomNavigationView.OnNavigationItemSelectedListener bottomNavListener
= new BottomNavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
        Fragment currentFragment = null;
        switch (menuItem.getItemId()) {
            case R.id.nav_map:
                //Map tab
                currentFragment = new MapFragment();
                break;
            case R.id.nav_searches:
                //Searches tab
                currentFragment = new SearchesFragment();
                break;
        }

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
currentFragment).commit();
        return true;
    }
};
```

Then for the “Map” tab on the listeners are declared for the buttons (example shown below) as well as listening for key presses in the EditText box for entering an address.

```
Button startSearch = getActivity().findViewById(R.id.startSearchButton);
startSearch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        showSearchInput();
    }
});
```

```
final EditText mapSearchBox = (EditText)
view.findViewById(R.id.map_search_bar);
mapSearchBox.setOnKeyListener(new View.OnKeyListener() { //Listen for key
presses from the search box at the top, if it is the "enter" key, take value
from the EditText and call another method
    @Override
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        if((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode ==
KeyEvent.KEYCODE_ENTER)) {
            addressToCoordinates(mapSearchBox.getText().toString()); //Run
addressToCoordinates with value from EditText as string
        }
        return false;
    }
});
```

To add the functionality for the user tapping on the map, the Mapbox library has a listener already setup for this purpose. In order to use it, the class implements the listener, the callback is defined when the map has loaded and then upon a click being registered, the method below runs.

```
@Override
public boolean onMapClick(@NonNull LatLng point) {
   if(markerView != null) {
       markerViewManager.removeMarker(markerView);
       View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder,
null);
       customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
       markerView = new MarkerView(point, customView);
       markerViewManager.addMarker(markerView);
       startLat = point.getLatitude();
       startLon = point.getLongitude();
       ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
       view.setVisibility(View.VISIBLE);
   } else {
       View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder,
null);
       customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
       markerView = new MarkerView(point, customView);
       markerViewManager.addMarker(markerView);
       startLat = point.getLatitude();
       startLon = point.getLongitude();
       ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
       view.setVisibility(View.VISIBLE);
   }
   return true;
}
```

This method checks if there currently is a marker or not, if there is then it removes it before adding another one.

After the map has finished loading, the user location is also plotted. This has a default time interval set which is a constant and is the time between each location update.

```
private void enableLocation(@NonNull Style mapStyle) {
   if(ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED)
{
       activity = (MainActivity) getContext(); //Essentially replacing "this"
as "this" can't be used in fragments as there is no context passed here
       LocationComponent locationComponent = mapboxMap.getLocationComponent();
       LocationComponentActivationOptions locationComponentActivationOptions =
LocationComponentActivationOptions.builder(activity,
mapStyle).useDefaultLocationEngine(false).build();

locationComponent.activateLocationComponent(locationComponentActivationOptions)
;
       locationComponent.setLocationComponentEnabled(true);
       locationComponent.setCameraMode(CameraMode.TRACKING);
       //locationComponent.setRenderMode(RenderMode.COMPASS);
       initLocationEngine();
   } else {
       requestPermissions(
               new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
               1
       ); //Request location permission if not already granted
   }
}
```

This method then has the possibility to invoke 1 of 2 methods depending on if permissions have been granted by the user. initLocation() will create a LocationEngine which will do all the location updating and requestPermissions will run the method below if the user has not allowed fine access location yet.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
   //Method called when requestPermissions(); gets called
   if(requestCode == 1) {
       if (permissions[0].equals(Manifest.permission.ACCESS_FINE_LOCATION) &&
grantResults[0] == PackageManager.PERMISSION_GRANTED) {
           //Check if location permissions have been granted
           mapboxMap.getStyle(new Style.OnStyleLoaded() {
               @Override
               public void onStyleLoaded(@NonNull Style style) {
                   enableLocation(style); //Rerun method, this time with
correct permissions
               }
           });
       }
   } else {
       Toast.makeText(getActivity(), R.string.location_access_not_permitted,
Toast.LENGTH_LONG).show(); //Let user know they need to allow location
permission
   }
}
```

This double checks the permissions and lets the user know if they need to do something in the form of a Toast.

Using the textbox present at the top of the screen, users should be able to input an address and the point will be plotted on the map. However the map library won't allow me to just input an address so I had to turn this into coordinates using an API call. I did this using an HTTP library and parsing the JSON response and then feeding the coordinates to the map library. All network operations must be done on a seperate thread (other than UI) to prevent skipping frames or lifecycle methods.

```
private void addressToCoordinates(@NonNull String address) {
   address = address.replaceAll("/[^a-zA-Z0-9 ]/g", "").replaceAll(" ", "%20");
   String url = "https://api.mapbox.com/geocoding/v5/mapbox.places/" + address
+ ".json?&access_token=" + getString(R.string.mapbox_token);
   //Variables must be declared before starting a thread as they have to be
final or effectively-final
   //Creating a thread - running network operations must happen on another
thread as the main/UI thread can skip frames or lifecycle methods
   Thread thread = new Thread(() -> {
       Thread.currentThread().setPriority(Thread.MIN_PRIORITY); //Important to
set the thread priority to MIN or at least less than main/UI thread to prevent
skipping frames or lifecycle methods
       OkHttpClient okHttpClient = new OkHttpClient(); //Initialising an
instance of the HTTP client
       Request request = new Request.Builder() //Build request with URL and
optional headers (not needed)
               .url(url)
               .build();
       Response response = null; //Cannot run query in a try(query here){} due
to a target API mismatch (so it runs on more devices)
       try {
           response = okHttpClient.newCall(request).execute();
           String responseStr = response.body().string();
           JSONObject jsonObject = new JSONObject(responseStr); //Convert the
string into a JSONObject for manipulation and data reading
           JSONArray allFeatures = jsonObject.getJSONArray("features");
           JSONObject firstObject = allFeatures.getJSONObject(0);
           JSONObject geometryObject = firstObject.getJSONObject("geometry");
           JSONArray coordPair = geometryObject.getJSONArray("coordinates");
//Final coordinate pair taken from API response in order to plot marker on the
map
           double lat = (Double) coordPair.get(1); //Has to be cast as double
           double lng = (Double) coordPair.get(0);
           startLat = lat;
           startLon = lng;
           getActivity().runOnUiThread(new Runnable() {
               @Override
               public void run() {
                   View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder,
null);
                   customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                   markerView = new MarkerView(new LatLng(lat, lng),
customView);
                   markerViewManager.addMarker(markerView);
                   ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
                   view.setVisibility(View.VISIBLE);
               }
           });
       } catch (JSONException e) {
           Toast.makeText(getActivity(), "Failed to plot point\nPlease try
```

To fetch current searches that have been stored in a Firestore database I used google's library.

```
private void getSearches() {
   onlineDatabase.collection("searches")
           .get()
           .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
               @Override
               public void onComplete(@NonNull Task<QuerySnapshot> task) {
                   ArrayList<CardModel> cards = new ArrayList<>();
                   if(task.isSuccessful()) {
                       for(QueryDocumentSnapshot documentSnapshot :
task.getResult()) {
                           CardModel cardModel = new CardModel();

cardModel.setSearchName(documentSnapshot.getString("searchName"));
                           cardModel.setSearchOwner("Owned by: " +
documentSnapshot.getString("owner"));
                           cards.add(cardModel);
                       }
                       cardAdapter = new CardAdapter(getActivity(), cards);
                       recyclerView.setAdapter(cardAdapter);
                   } else {
                       Toast.makeText(getContext(), "Failed to load
searches\nPlease try again", Toast.LENGTH_LONG).show();
                   }
               }
           });
}
```

This listens to updates from the databases and applies them to a cardview model, which can be seen in the final code section.

In order for the application to know if it needs to plot an isochrone or not, I created a boolean variable which gets changed when the map view is reinitialised. If it is true then the application runs the code to make an isochrone, if not then it just carries on and lets the user place markers and start searches. In order to do this, I have a private variable in the MapFragment which gets updated in the onViewCreated() method where it checks the bundle arguments.

```
startBundle = this.getArguments();
if(startBundle != null) {
   polygonToPlot = startBundle.getBoolean("polygonToPlot", false);
}
```

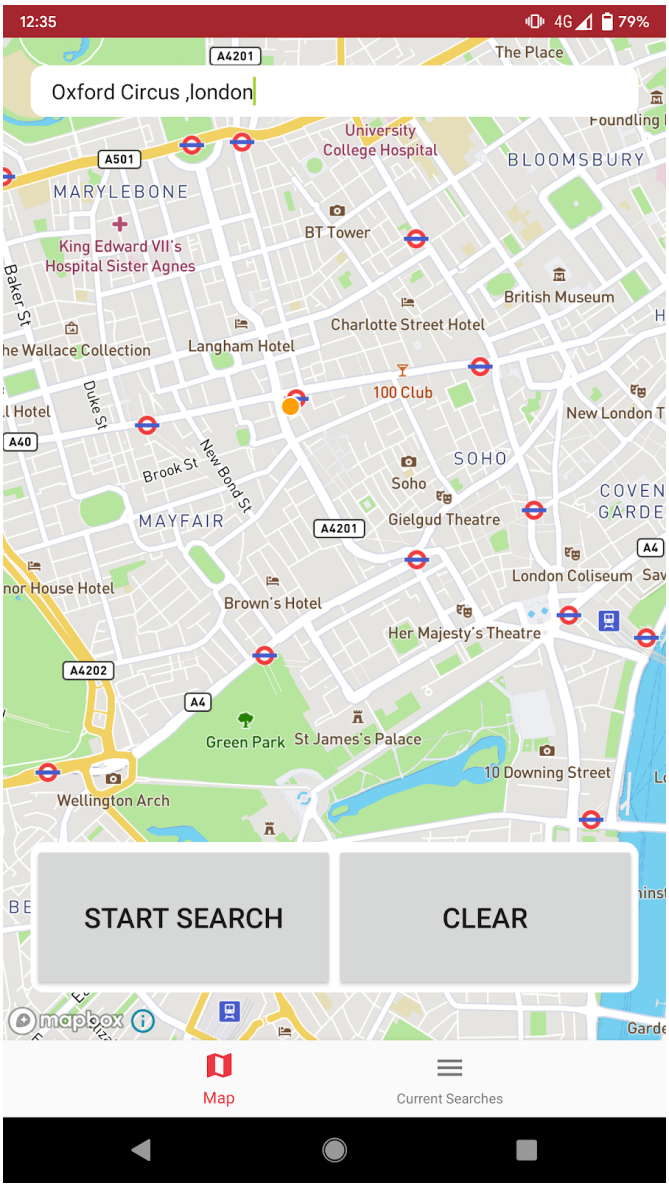
Then in the onMapReady() method I check the value of polygonToPlot to see if the isochrone logic needs to be run or not.

```
if(polygonToPlot) {
   // Run isochrone code here
}
```

Testing

I tested each function after making it. Respective tests can be found below.

Testing the address function:



This function works, as you can see the orange location dot over the Oxford Circus Underground station, the start search rectangle also shows up and the buttons work to present the inputs for the search. However when zooming out, the location point is offset slightly from the actual coordinates, this is due to the fact that adding markers normally on the map was deprecated and they now have to be added through layers on the map. In order to plot the point, I used the new methods available in the library although it was designed for adding text next to a location, although the location of the point is still exactly where the user tapped, just the marker itself is slightly offset. I did carried out some more

tests, such as inputting nothing into the textbox and just pressing enter, this presents the user with a Toast saying they need to enter something into the box.

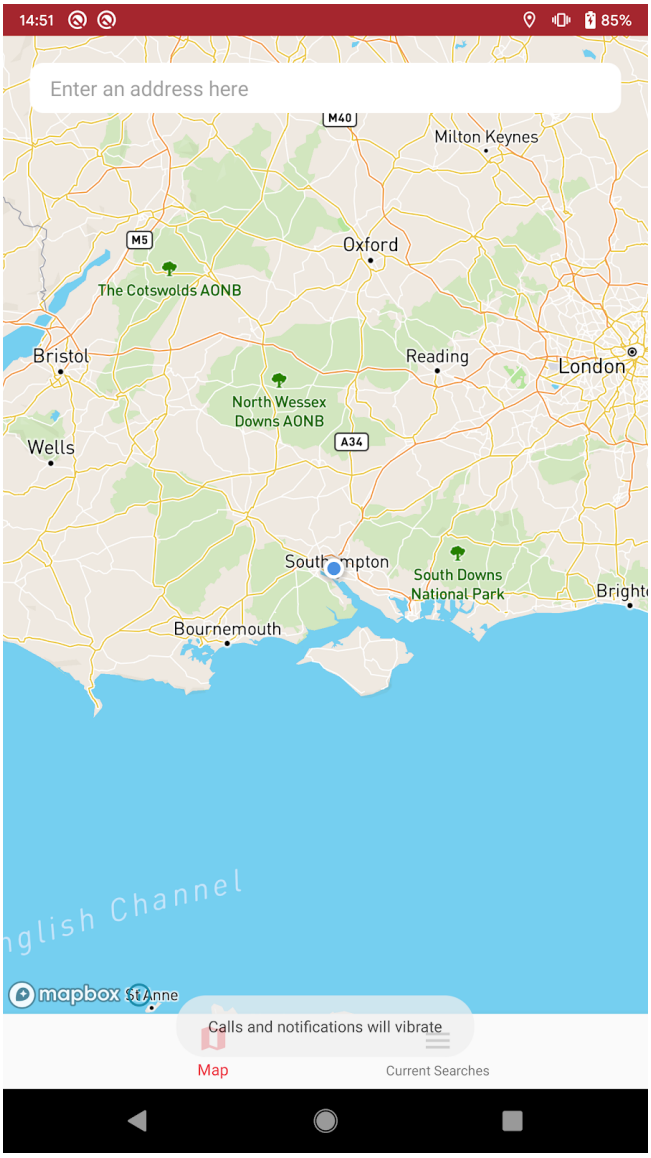
```
if(mapSearchBox.getText().toString().length() > 0) {
    addressToCoordinates(mapSearchBox.getText().toString());
} else {
    Toast.makeText(getActivity(), "Please enter something into the search box", Toast.LENGTH_SHORT).show();
}
```

I then also found that certain places in the world, such as Niger, would produce and exception due to the API returning an integer. It would cause a ClassCastException as shown below.

```
E/AndroidRuntime: FATAL EXCEPTION: Thread-4
        Process: com.bengavin.missingpersons, PID: 7752
        java.lang.ClassCastException: java.lang.Integer cannot be cast to java.lang.Double
        at com.bengavin.missingpersons.MapFragment.lambda$addressToCoordinates$0$MapFragment(MapFragment.java:320)
```

This was being thrown because coordPair is an Object so in order to fix this I had to change two lines.

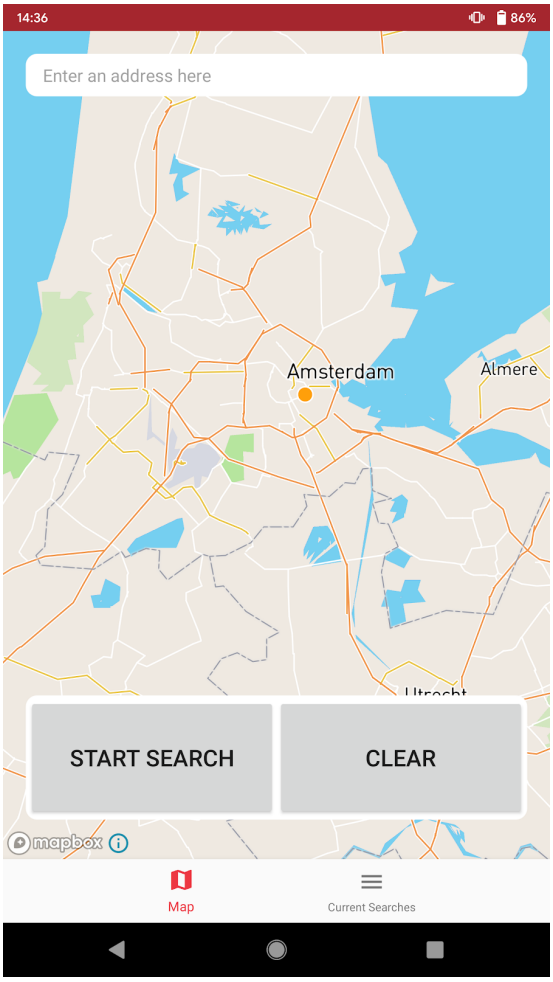
```
double lat = (double) coordPair.get(1);
double lng = (double) coordPair.get(0);
//Above is what would cause the exception, so to get around this I used .doubleValue();
double lat = ((Number) coordPair.get(1)).doubleValue();
double lng = ((Number) coordPair.get(0)).doubleValue();
```



Next I tested the current location function in the app. This works well, the location updates when the user moves around. An issue occurred during testing where using render mode would cause an exception to be thrown by the library.

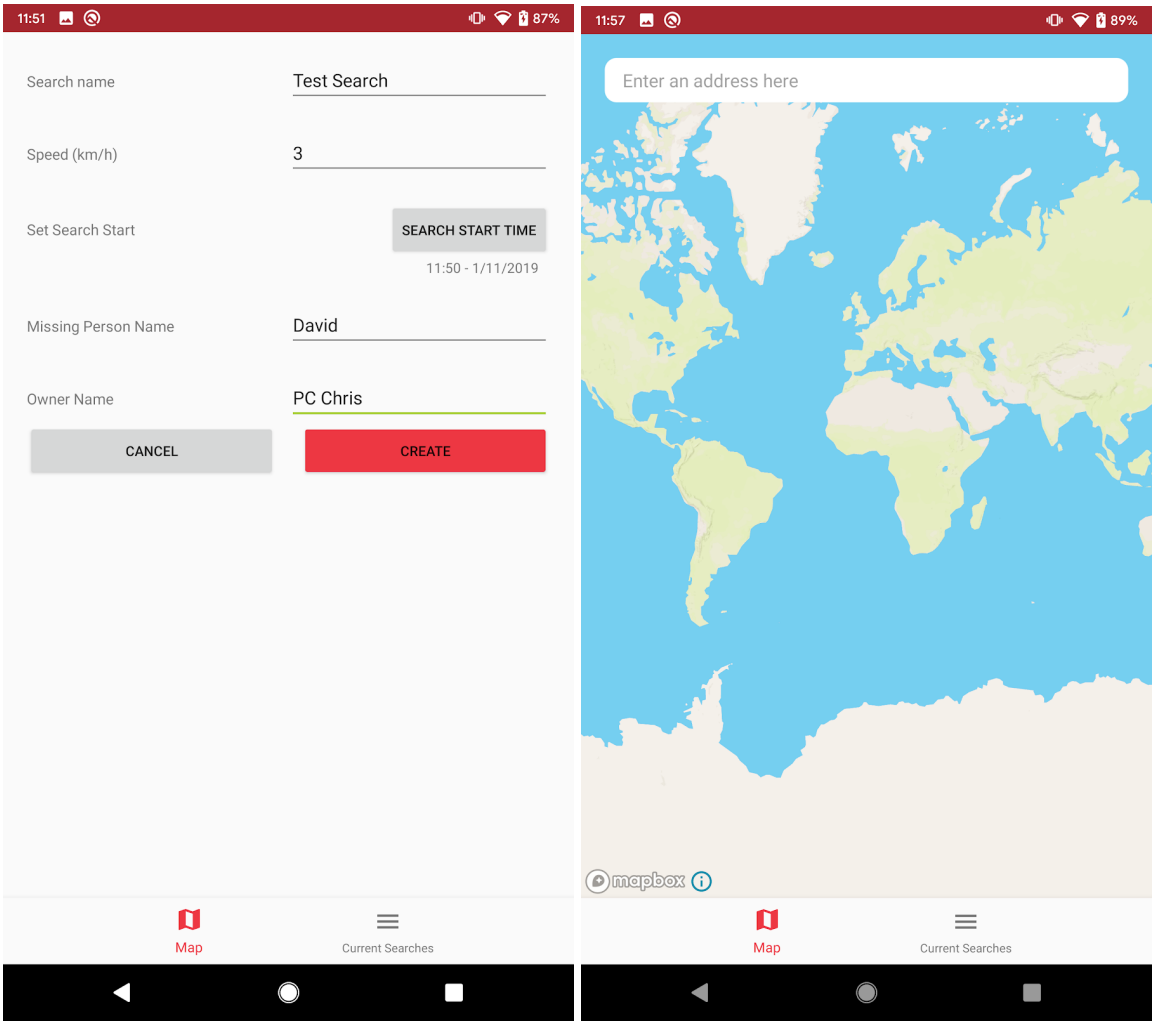
```
Fatal Exception: java.lang.IllegalStateException: Calling getSourceAs when a newer style is loading/has loaded.
    at com.mapbox.mapboxsdk.maps.Style.validateState(Style.java:522)
    at com.mapbox.mapboxsdk.maps.Style.getSourceAs(Style.java:132)
```

This was referenced by someone in a Github issue (https://github.com/mapbox/mapbox-gl-native/issues/14889) which was claimed to be fixed in v8.5.0 although never was, as more testing showed that the error still occurred, so I had to remove the render mode for now (small arrow on location point showing compass direction).

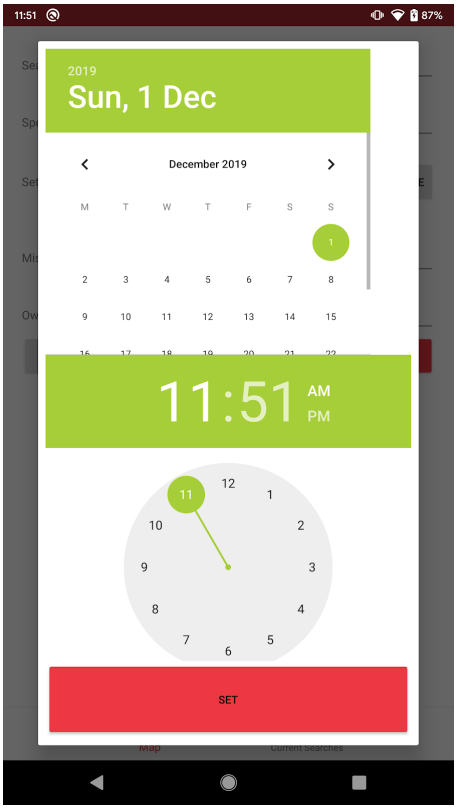


Next I tested tapping a location on the map. This worked perfectly, as shown above. However the issue with the offset still exists (just as it did in the address to coordinate testing), but it still works.

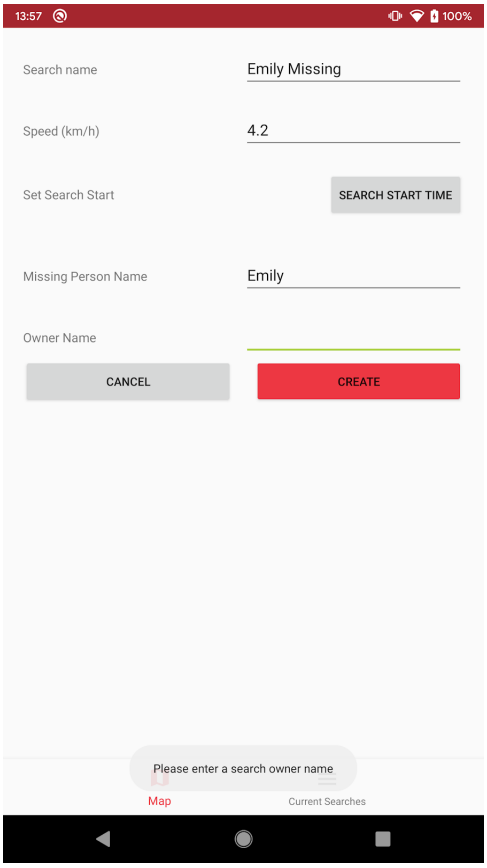
Pressing the start search button brings up the new search screen with all the inputs, and the clear button also works as it removes all current markers on the map and also gets rid of the box with the start search and clear button in it.



Clicking “Search Start Time” brings up an AlertDialog with a date and time picker in them, it then returns the set time and puts it below the button so that the user can see the time went through.



Next I tested the input boxes with data to check they do the right thing with it. If no value is entered into a textbox a toast is made alerting the user to input something into the textbox (see below).



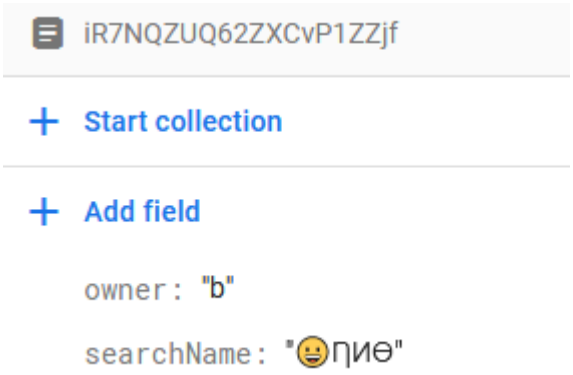
If the search start time is empty, it gets set to the current system time.

```
try {
   speedInputValue = Float.parseFloat(speedInput.getText().toString());
   if(searchNameInput.length() == 0 || missingPersonNameInput.length() == 0 || ownerNameInput.length() == 0) {
       Toast.makeText(getActivity(), "Please fill in all of the fields", Toast.LENGTH_SHORT).show();
       if (time == 0) {
           time = System.currentTimeMillis();
       }
   } else {
       createSearch(searchNameInput, ownerNameInput, bundle.getDouble("lat", 0), bundle.getDouble("lon", 0),
speedInputValue, missingPersonNameInput, time);
   }
} catch (NumberFormatException e) {
   Toast.makeText(getActivity(), "Please enter a speed", Toast.LENGTH_SHORT).show();
}
```

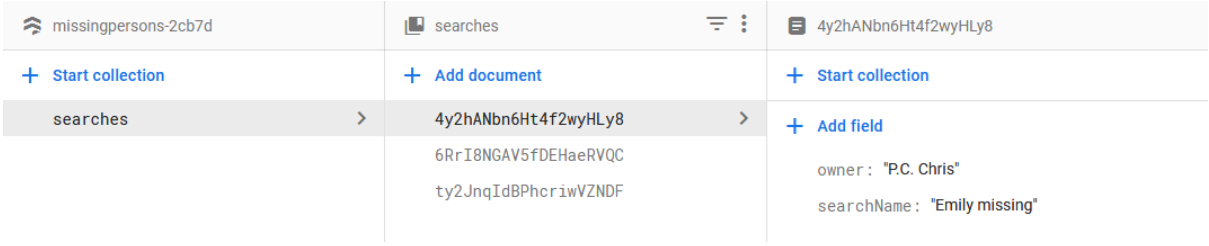
I then tested to see if the data below caused any problems within the application, and recorded what happened. The data included normal strings which is likely to be what the user will input to including emojis and unicode characters which is less likely to be an input, but it is testing to see how the app will handle it. I do expect all of them to continue as normal, as the text variables are all strings so Java will store it as a string and won't try and interpret it.

Test data	**Expected result**	**Actual result**
test search	Continues as normal	Continues as normal

James’ search	Continues as normal	Continues as normal
@#3482<<//;search|	Continues as normal	Continues as normal
😬ηиθ	Continues as normal	Continues as normal



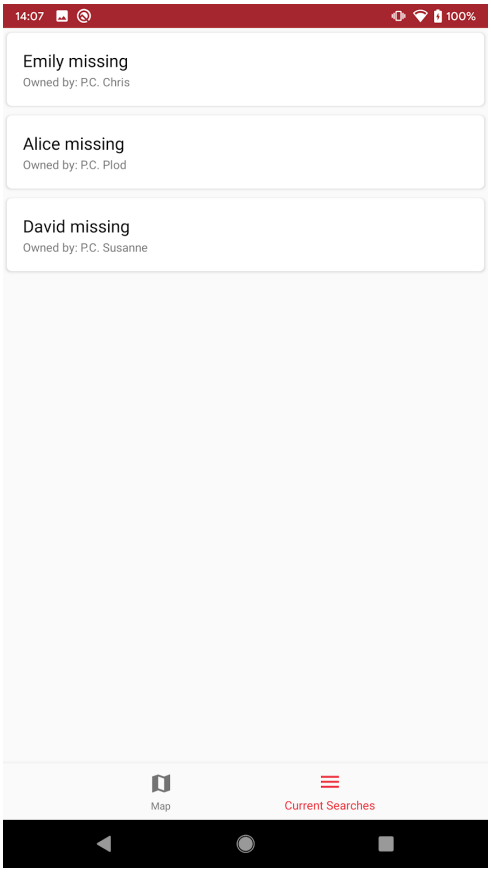
On the searches tab, users can see current searches that are fetched from an online database. To test this I added a few test searches into the database and observed the results.



This is the database schema. Each document under the “searches” collection has certain attributes to it.

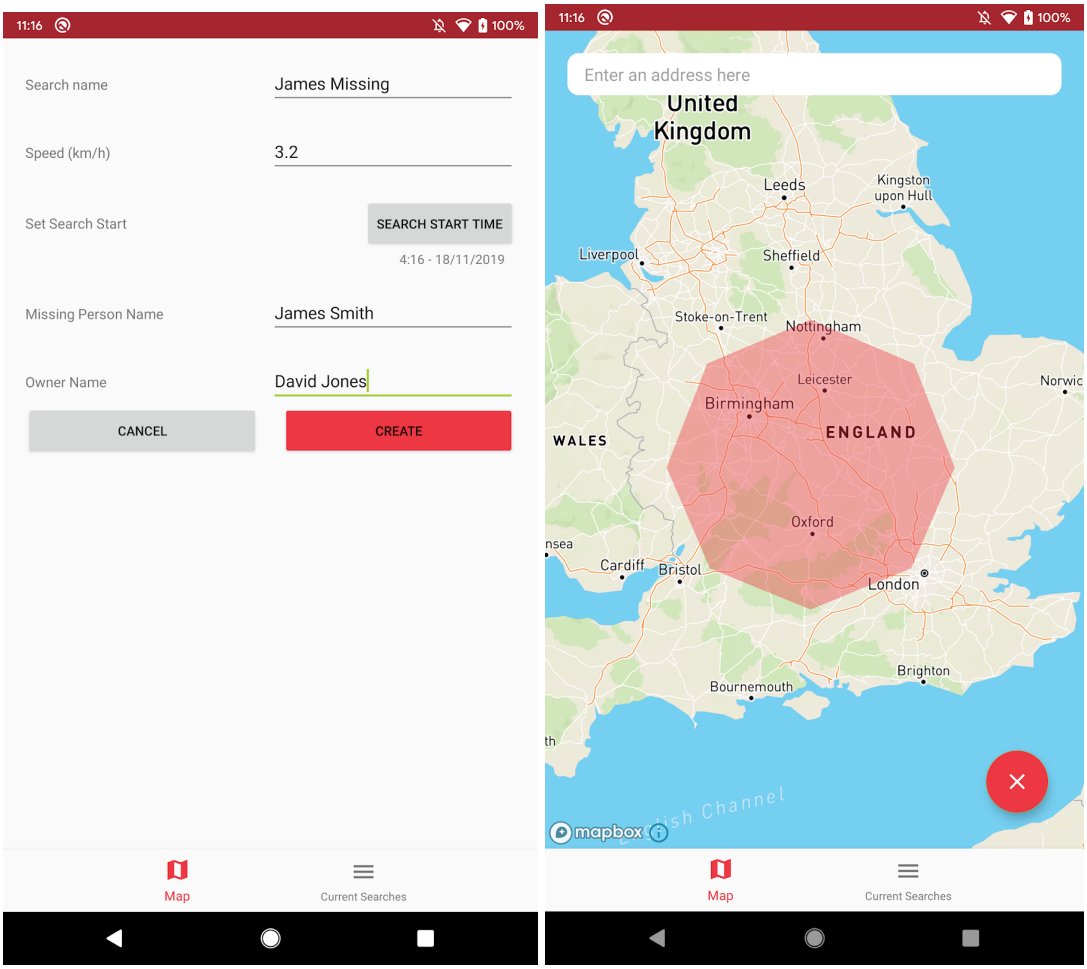
Then using the following code, each document should create a CardView to hold each search in it.

```
private void getSearches() {
    onlineDatabase.collection("searches")
            .get()
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    ArrayList<CardModel> cards = new ArrayList<>();
                    if(task.isSuccessful()) {
                        for(QueryDocumentSnapshot documentSnapshot : task.getResult()) {
                            CardModel cardModel = new CardModel();
                            cardModel.setSearchName(documentSnapshot.getString("searchName"));
                            cardModel.setSearchOwner("Owned by: " +
documentSnapshot.getString("owner"));
                            cards.add(cardModel);
                        }
                        cardAdapter = new CardAdapter(getActivity(), cards);
                        recyclerView.setAdapter(cardAdapter);
                    } else {
                        Toast.makeText(getContext(), "Failed to load searches\nPlease try again",
Toast.LENGTH_LONG).show();
                    }
                }
            });
}
```



All of the searches show up correctly, so this function of the app works fine.

Lastly I tested the plotting of a simple search area, which should find the maximum possible distance the person could've have reached at the speed input, constantly. To test this I input some test data and this is what the app did.



This appears to be correct as if “James” was to walk at 3.2km/h for ~ 1 day and 7 hours (as at the time of starting the search it was 11am 19/11/19) of a he would have walked ~100km.

Review

What was done?

In this iteration I have made and tested plotting a basic isochrone, using the search bar to enter an address, tapping on the screen to add a marker, inputting data on the new search screen, live location on a map and current searches can also be viewed.

How was it tested?

After adding major changes to the GUI, I would test it to make sure that none of the elements overlapped or where in bad places. Inputs were tested, I tested with potentially destructive data and also testing with nothing in the box. All of the buttons were tested too, and worked fine. I also tested each method after making it to make sure it worked properly, of which can be seen above.

Success Criteria Met

- 1 - Interactive map

- 2 - User location on map
- 3 - Use an address to plot a marker
- 4 - Tap the map to plot a marker
- 5 - Simple design
- 6 - Search parameter input
- 10 - Current searches
- 7 - Isochrone algorithm - This has been partially done in this iteration, although applying topography will come in a later iteration

Iteration 2 - Writing new search to database and search insights

For this iteration I created a search insight function to show more details about the missing person and then have the ability to open the map in order to see the search area. In this iteration, when creating a new search it will be uploaded to the database as well.

Code

All of the code changes will be noted in the Final Code section.

To start off, I implemented saving the new search that can be created to the database, this is rather crucial as it is what allows the user to reopen their search after they have closed their app, it also will allow other users to see the search boundaries. This was fairly straight forward to implement, all it required was a hashmap as the storage and then a database call was made.

```
private void createSearch(String searchName, String ownerName, double startLat, double startLon, float speed,
String missingPersonName, long time) {
   Map<String, Object> data = new HashMap<>();
   data.put("searchName", searchName);
   data.put("owner", ownerName);
   data.put("startLat", startLat);
   data.put("startLon", startLon);
   data.put("speed", speed);
   data.put("missingPersonName", missingPersonName);
   data.put("startTime", time);

   onlineDatabase.collection("searches")
           .add(data)
           .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
               @Override
               public void onSuccess(DocumentReference documentReference) {
                   Toast.makeText(getActivity(), "Search created and saved to online database",
Toast.LENGTH_SHORT).show();
                   MapFragment mapFragment = new MapFragment();
                   Bundle newBundle = new Bundle();
                   newBundle.putBoolean("polygonToPlot", true);
                   newBundle.putFloat("speedInputValue", speed);
                   newBundle.putLong("startTime", time);
                   newBundle.putDouble("lat", lat);
                   newBundle.putDouble("lon", lon);
                   mapFragment.setArguments(newBundle);
                   getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
               }
           })
           .addOnFailureListener(new OnFailureListener() {
               @Override
               public void onFailure(@NonNull Exception e) {
                   Toast.makeText(getActivity(), "Search created but failed to save to online database",
Toast.LENGTH_SHORT).show();
               }
           });
}
```

I updated NewSearchFragment.createSearch to now implement adding in a database call, on completion the search is then made.

Next I created a simple screen that just allowed the user to see more information about the search/missing person and it would allow them to open the map and see the search area. The easiest way to get this information was to just fetch that specific search from the database.

```
private void fetchData(String uid, FirestoreDataCallback firestoreDataCallback) {
   onlineDatabase.collection("searches")
           .document(uid)
           .get()
           .addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
               @Override
               public void onSuccess(DocumentSnapshot documentSnapshot) {
                   String searchName = documentSnapshot.getString("searchName");
                   String owner = documentSnapshot.getString("owner");
                   String missingPersonName = documentSnapshot.getString("missingPersonName");
                   String speed = String.valueOf(documentSnapshot.get("speed"));
                   String startLat = String.valueOf(documentSnapshot.get("startLat"));
                   String startLon = String.valueOf(documentSnapshot.get("startLon"));
                   String startTime = String.valueOf(documentSnapshot.get("startTime"));
                   firestoreDataCallback.onData(searchName, owner, missingPersonName, speed,
startLat, startLon, startTime);
               }
           });
}
```

This function uses a “uid” which is autogenerated by Firebase when the search is created, this is fetched when all of the searches come back from the current searches screen, but is

then passed to this screen via a bundle. This screen also had two buttons, one which returned the user to the current searches screen and the other would open the map.

```
openMapButton.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       MapFragment mapFragment = new MapFragment();
       Bundle newBundle = new Bundle();
       newBundle.putBoolean("polygonToPlot", true);
       newBundle.putFloat("speedInputValue", speed);
       newBundle.putLong("startTime", time);
       newBundle.putDouble("lat", startLat);
       newBundle.putDouble("lon", startLon);
       mapFragment.setArguments(newBundle);
       getFragmentManager().beginTransaction().replace(R.id.fragment_container, mapFragment).commit();
   }
});
```

I also put the time format back into something interpretable (instead of unix), I used SimpleDateFormat to do this.

```
Date date = new Date(Long.valueOf(startTime));
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("HH:mm dd-MM-yyyy z", Locale.getDefault());
String dateString = simpleDateFormat.format(date);
```

However, in order to bring up this screen I had to make my own onClickListener for the CardView. This required an interface and some modifications to the CardHolder and CardAdapter.

```
package com.bengavin.missingpersons;

import android.view.View;

public interface CardClickListener {

   void onCardClickListener(View view, int position);

}
```

CardAdapter

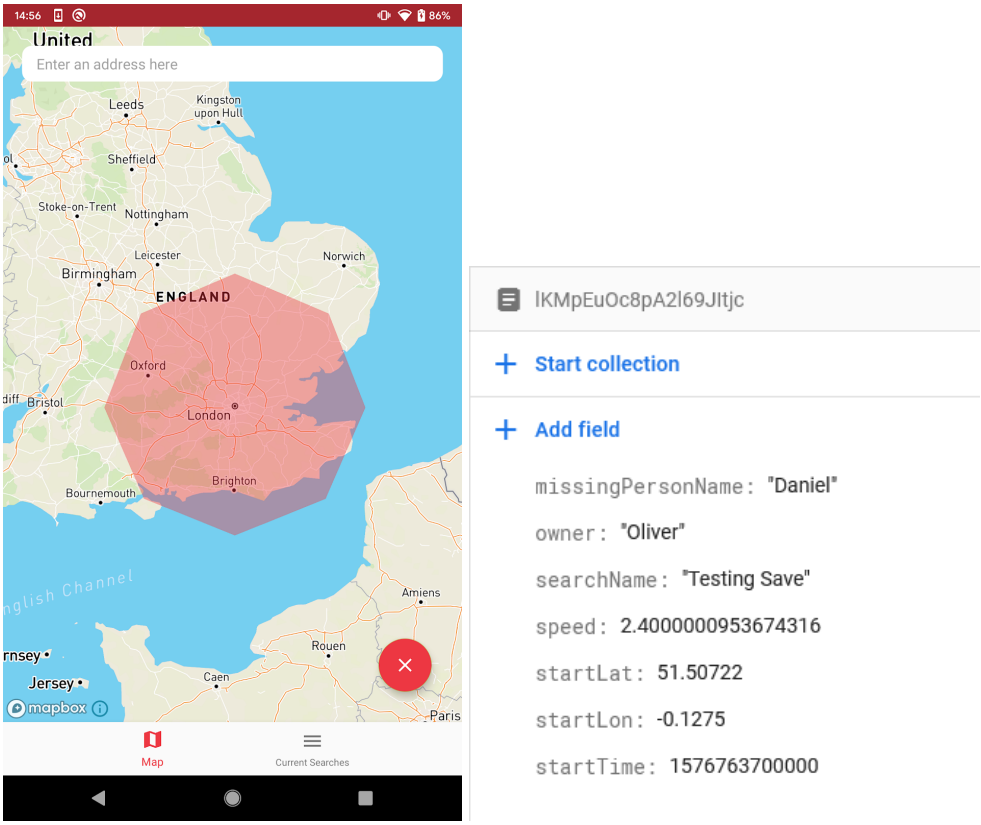
```
holder.setCardClickListener(new CardClickListener() {
   //Use interface to make a click listener
   @Override
   public void onCardClickListener(View view, int position) {
       String name = cardModels.get(position).getSearchName();
       String owner = cardModels.get(position).getSearchOwner().substring(10);
       String uid = cardModels.get(position).getUid();
       //Get attributes from the clicked search card

       Fragment fragment = new SearchInsightFragment();
       Bundle bundle = new Bundle();
       bundle.putString("name", name);
       bundle.putString("owner", owner);
       bundle.putString("uid", uid);
       fragment.setArguments(bundle);
       //This makes a new fragment and puts the attributes retrieved above, into a bundle which can
be used by the secondary fragment

       FragmentManager fragmentManager = ((AppCompatActivity)context).getSupportFragmentManager();
       fragmentManager.beginTransaction().replace(R.id.fragment_container,
fragment).addToBackStack(null).commit(); //Change fragments
   }
});
```

CardHolder

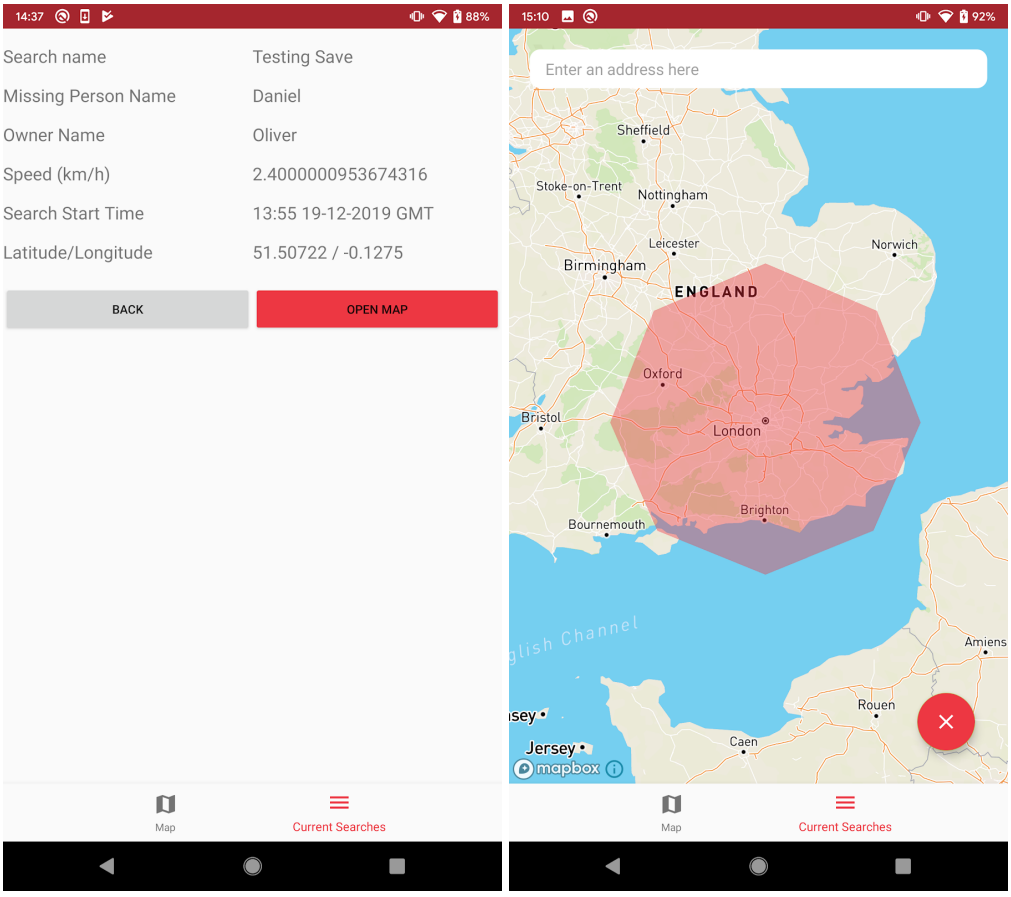
```
CardHolder(@NonNull View itemView) {
   super(itemView);

   this.mSearchName = itemView.findViewById(R.id.searchName);
   this.mSearchOwner = itemView.findViewById(R.id.searchOwner);

   itemView.setOnClickListener(this);
}

@Override
public void onClick(View v) {
   this.cardClickListener.onCardClickListener(v, getLayoutPosition());
}

public void setCardClickListener(CardClickListener cardClickListener) {
   this.cardClickListener = cardClickListener;
}
```

Testing

I tested the function to allow the user to save a search the database first. I expect this to create the search, display it on screen and then also save it on the cloud database.



This successfully created the search and saved it to the database. Next I tested the search insight functionality.



This worked as expected and upon pressing the open map button, it opens a map and plots the updated version of the isochrone.

There was nothing further testing to be done in this section as there were no inputs added.

Review

What was done?

In iteration 2 I have implemented the ability to save the searches to an online database and also made a “search insight” screen which allows users to see more information about the search.

How was it tested?

I made sure to test the GUI layouts whilst making them, like in the previous iteration, just to check everything looked good. I then made sure to test the database save function by looking at the database to make sure that it had been written to with the correct information. Then the search insight was tested, where I compared the information which was just written to the database and made sure it matched with what was on the insight page.

Success Criteria Met

- 5 - Simple design

- 11 - Search Insight
- 12 - Saving searches to an online database

Iteration 3 - Isochrone with Topography

For this iteration I plan to apply elevation data of surroundings to the search boundary and adjust it accordingly.

Code

The first thing that had to be done was to fetch all of the elevations from the points calculated in iteration 1 by the Isochrone.getMaxCoordinates() method. To do this I made a HTTP request to a Mapbox API which would return a JSON response. Once again this had to go in a thread because network operations should not be done on the main/UI thread. I used a for loop to loop through all of the coordinates in the list but because the API endpoint wanted lon lat instead of lat lon so I used some extra counters to track the position of those.

```
for (int i = 0; i < 9; i++) {
                String url = "https://api.mapbox.com/v4/mapbox.mapbox-terrain-v2/tilequery/" +
coordinates.get(k) + "," + coordinates.get(j) + ".json?layers=contour&limit=50&access_token=" +
context.getString(R.string.mapbox_token);
                Thread.currentThread().setPriority(Thread.MIN_PRIORITY);
                OkHttpClient okHttpClient = new OkHttpClient();
                Request request = new Request.Builder()
                        .url(url)
                        .build();
                Response response = null;
                try {
                    response = okHttpClient.newCall(request).execute();
                    String responseStr = response.body().string();
                    JSONObject jsonObject = new JSONObject(responseStr);
                    JSONArray allFeatures = jsonObject.getJSONArray("features");
                    ArrayList<Integer> elevations = new ArrayList<>();
                    for (int m = 0; m < allFeatures.length(); m++) {
                        JSONObject feature = allFeatures.getJSONObject(m);
                        JSONObject properties = feature.getJSONObject("properties");
                        int featureElevation = (Integer) properties.get("ele"); // API always returns an
int
                        elevations.add(featureElevation);
                    }
                    int highestElevation = 0;
                    try {
                        highestElevation = Collections.max(elevations); // Get max elevation
                    } catch (NoSuchElementException e) {
                        Toast.makeText(context, "The elevation for point " + i + " failed to load - using
max possible distance", Toast.LENGTH_SHORT).show();
                    }
                    elevationsAtPoints.add(highestElevation);
                } catch (JSONException e) {
                    Toast.makeText(context, "Failed to plot point\nPlease try again later",
Toast.LENGTH_LONG).show();
                } catch (IOException e) {
                    Toast.makeText(context, context.getString(R.string.ioexception_error_message),
Toast.LENGTH_LONG).show();
                } finally {
                    if(response != null) {
                        response.body().close();
                    }
                }
                j = j + 2;
                k = k + 2;
                if(i == 8) {
                    tilequeryCallback.onData(elevationsAtPoints);
                }
            }
```

During developing this section of code, I had to fix the fact that I needed to make android OS calls from a non activity class (toasts so the user knows if something went wrong) which I solved by the use of a constructor using android Context in order to show messages.

```
private Context context;
public Isochrone(Context context) {
    this.context = context;
}
```

I also had to create a callback for this operation so that the code to plot the polygon wouldn't run before the elevations had been fetched and processed. An array list needed to be returned containing all of the elevations so that was the only parameter for the callback.

```
public interface TilequeryCallback {

    void onData(ArrayList<Integer> elevationAtPoints);

}
```

After fetching the elevations, they then needed to be applied to each of the points calculated earlier. The way I work out how far to retract the points on the isochrone is by the rule of "for every 10m of elevation gained or lost, add a minute". To do this, I needed to first use the height at the start point and then use the height at each of the outer points to get a difference, this difference would then be divided by 10 to get the number of minutes to subtract from each point. A distance is calculated by Isochrone.getDistanceTravelled with the speed given by the owner of the search.

```
public ArrayList<Double> applyElevationsToPoints(ArrayList<Double>
pointCoordinates, ArrayList<Integer> elevations, double speed) {
    int j = 0;
    int k = 1;
    int elevationAtStartPoint = elevations.get(elevations.size()-1);
    ArrayList<Double> endCoordinates = new ArrayList<>();
    for(int i = 0; i < pointCoordinates.size()/2; i++) {
        int bearing = 0;
        double lat = pointCoordinates.get(j);
        double lon = pointCoordinates.get(k);
        int elevationAtPoint = elevations.get(i);
        int heightDifference = elevationAtStartPoint - elevationAtPoint;
        int minutesToAdd = Math.abs(Math.round(heightDifference /
HEIGHT_PER_MINUTE));
        double distanceTravelled = getDistanceTravelled(minutesToAdd * 60,
speed);
        ArrayList<Double> retractionCoordinates = retractDistance(bearing, lat,
lon, distanceTravelled);
        double endLat = retractionCoordinates.get(0);
        endCoordinates.add(endLat);
        double endLon = retractionCoordinates.get(1);
        endCoordinates.add(endLon);
        j = j + 2;
        k = k + 2;
        bearing = bearing + 45;
    }
    return endCoordinates;
}
```

Then I made a retractDistance method which inverts the original bearing to face it in the opposite direction, sets the starting point to the outer point and goes X distance calculated by the getDistanceTravelled before to get an end point, this gets put back into an array list which is returned at the end.

```
private ArrayList<Double> retractDistance(int bearing, double lat, double lon, double distance) {
    lat = toRadians(lat);
    lon = toRadians(lon);
    double oppositeBearing = toRadians(360 - bearing); // Invert bearing/direction
    double distanceRatio = distance/6371.01;
    ArrayList<Double> endCoordinates = new ArrayList<>();
    double endLat = Math.asin(Math.sin(lat)*Math.cos(distanceRatio) +
Math.cos(lat)*Math.sin(distanceRatio)*Math.cos(oppositeBearing));
    endCoordinates.add(toDegrees(endLat));
    double endLon = lon + Math.atan2(Math.sin(oppositeBearing)*Math.sin(distanceRatio)*Math.cos(lat),
Math.cos(distanceRatio)-Math.sin(lat)*Math.sin(endLat));
    endCoordinates.add(toDegrees(endLon));
    return endCoordinates;
}
```

Testing

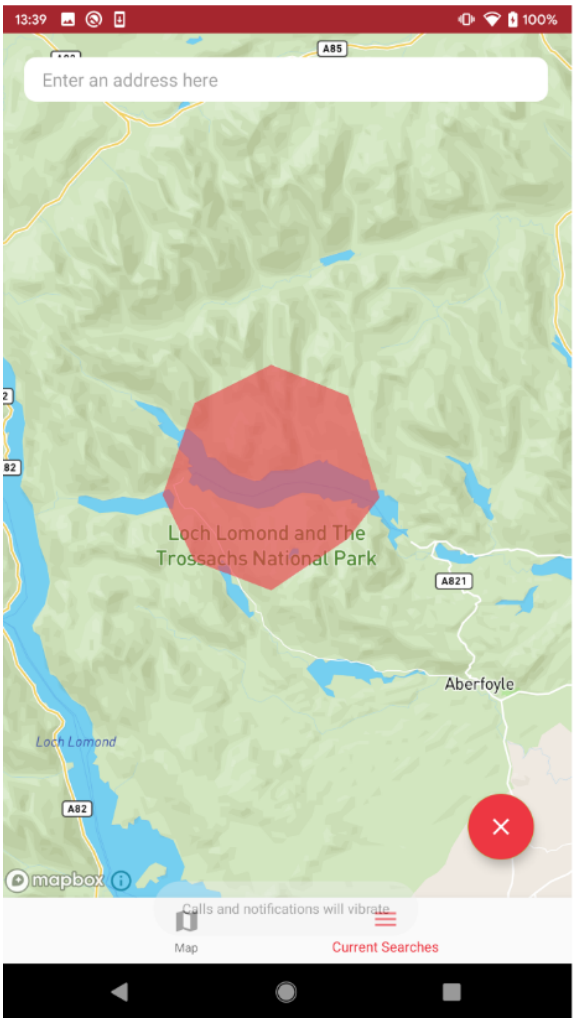
I first tested fetching heights with the API request. To do this I have to call the function with a new instance of the callback as one of the parameters.

```
isochrone.getElevations(coordinates, new TilequeryCallback() {
    @Override
    public void onData(ArrayList<Integer> elevationAtPoints) {
        Log.i("app", elevationAtPoints.toString));
    }
});
```

After running this, the app logged elevationAtPoints as shown below.

```
[380, 650, 530, 490, 210, 200, 190, 530, 110]
```

To test the topography adjustments made to the isochrone now, I created a new search in a mountainous area in order to highlight the effect that this has on the search area.



This worked successfully as it has retracted parts of the isochrone where there were sharp changes in altitude.

Review

What was done?

In this iteration I have added topography to the isochrone to make the search area slightly more accurate.

How was it tested?

It was tested by making sure that the isochrone has visually changed shape to account for the steeper incline/decline in the terrain which is also visible on the map.

Success Criteria Met

- 5 - Simple Design
- 7 - Isochrone Algorithm - Fully completed
- 8 - Fetching and applying elevation data

Iteration 4 - Deleting Searches

In iteration 4, I plan to add the ability for the owner of the search to delete it, this will remove it from the online database and any cached versions on any devices with the application installed. It will use a button which will bring up an alert to confirm whether the user actually wants to delete the search or not.

Code

First of all I had to make sure that when any new searches got written to the database, they included an ownerID which is a unique ID for each device used. This UID is fetched from the android OS (https://developer.android.com/reference/android/provider/Settings.Secure.html#ANDROID_ID), this value is unique enough for the database, it can be changed/spoofed by rooted phones and also changes if the app is reinstalled on Android 8.0 or lower. However for the sake of this project, this is fine as it's highly likely users will not be using a rooted phone and OS version > 8.0.

```
String androidUID = Settings.Secure.getString(getContext().getContentResolver(), Settings.Secure.ANDROID_ID);
```

This UID gets put in the HashMap along with the rest of the data and then it is updated in the database.

```
Map<String, Object> data = new HashMap<>();
data.put("ownerID", androidUID);
...
```

Then when fetching the searches in SearchInsightFragment.java the UID is fetched from the search. Then if this UID is equal to the one stored on the device (Settings.Secure.ANDROID_ID) then it will show the button on the screen.

```
if(androidUID.equals(ownerID)) {
   Button deleteButton = view.findViewById(R.id.insightDeleteButton);
   deleteButton.setVisibility(View.VISIBLE);
   deleteButton.setOnClickListener(...)
}
```

There is no need for an else statement as the user does not need to be alerted if they are not the owner, the button will just stay in the “gone” state so they can’t see or click it.

Inside the onClickListener for the delete button, I then put an alert dialog in so that the user has to confirm if they want to delete the search before it is gone forever.

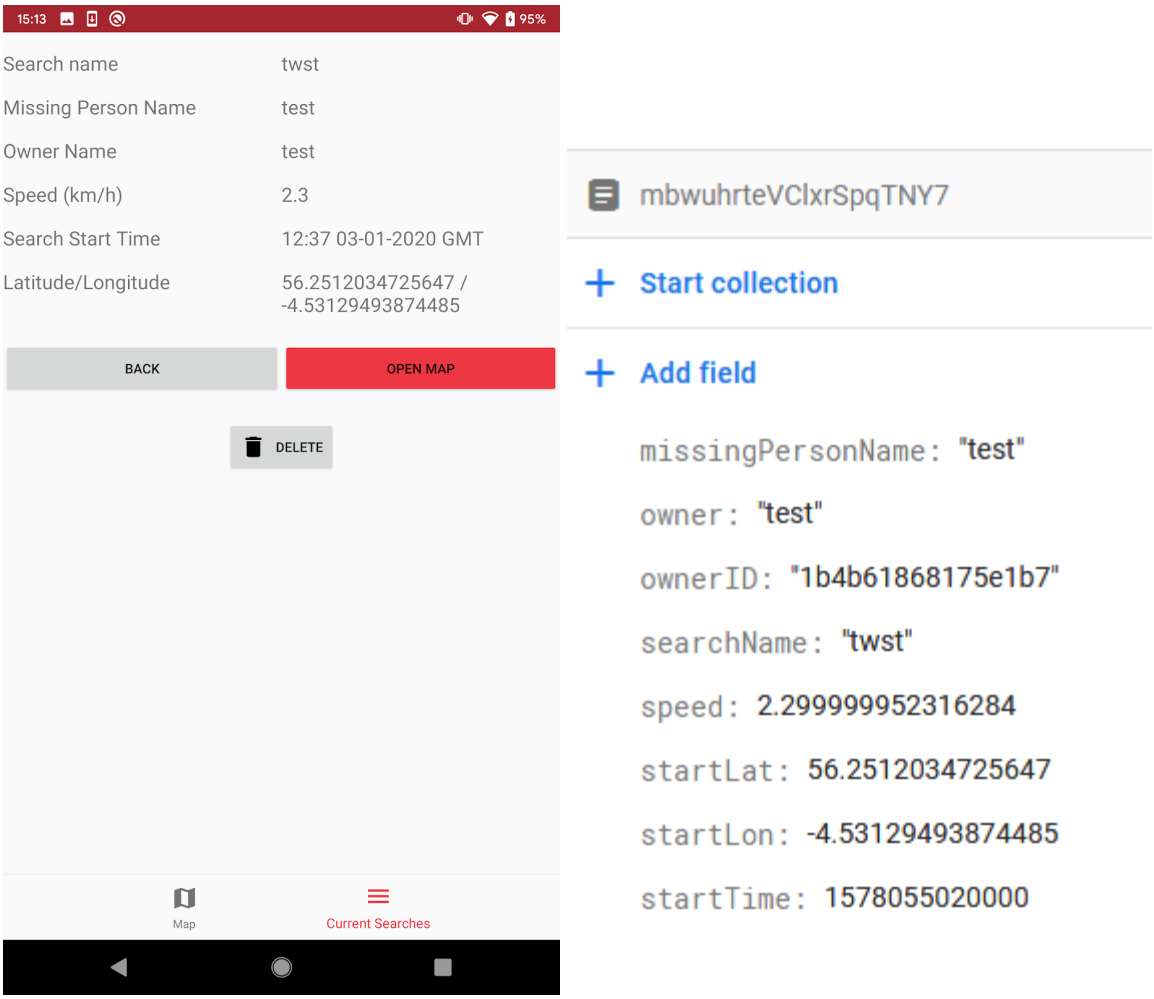
```
deleteButton.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       new AlertDialog.Builder(getContext())
               .setTitle("Confirm action")
               .setMessage("Are you sure you want to delete this search? This action cannot be undone.")
               .setIcon(R.drawable.ic_error_black_24dp)
               .setPositiveButton()
               .setNegativeButton()
               .show();
});
```

The negative button uses the “no” string resource and has no button click listener action attached to it so it just closes the dialog. The positive button has the code which will delete the search from the database using the unique ID for the search (generated by firebase).

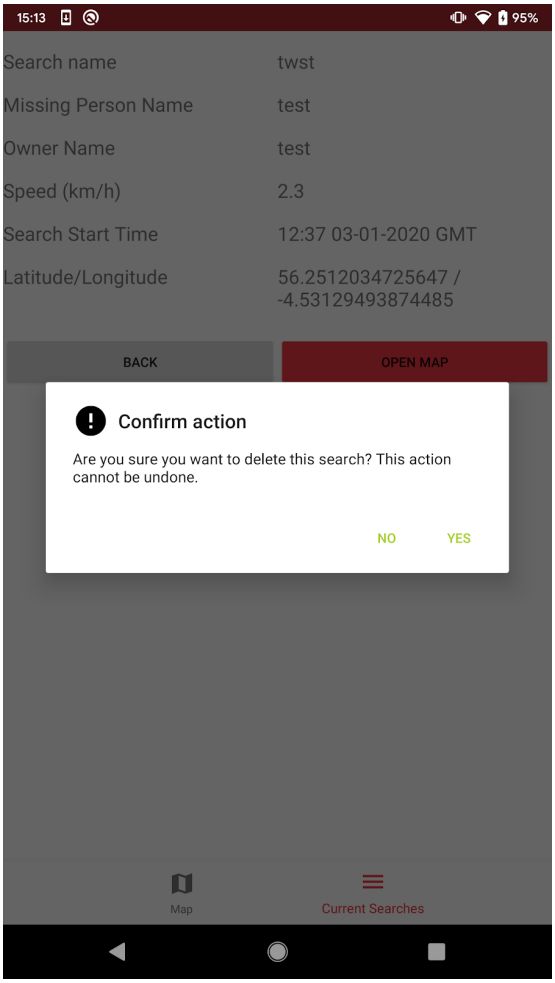
```
.setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
   @Override
   public void onClick(DialogInterface dialog, int which) {
       onlineDatabase.collection("searches")
               .document(documentUID)
               .delete()
               .addOnSuccessListener(new OnSuccessListener<Void>() {
                   @Override
                   public void onSuccess(Void aVoid) {
                       Toast.makeText(getActivity(), "Successfully deleted search",
Toast.LENGTH_SHORT).show();
                       getFragmentManager().popBackStackImmediate();
                   }
               })
               .addOnFailureListener(new OnFailureListener() {
                   @Override
                   public void onFailure(@NonNull Exception e) {
                       Toast.makeText(getActivity(), "Failed to delete search\nPlease try
again later", Toast.LENGTH_SHORT).show();
                   }
               });
   }
```

Testing

First, I tested whether the delete button shows up when it should. I first created a search on the device used for testing. This should make the delete button appear on the search insight screen.

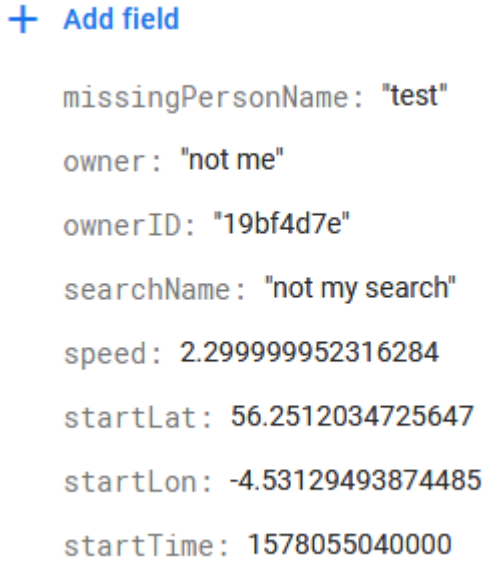


This function works, the delete button is visible on the owners device as seen on the left. On the right is what is seen in the database, there is an ownerID which is unique to the device which means this works too.

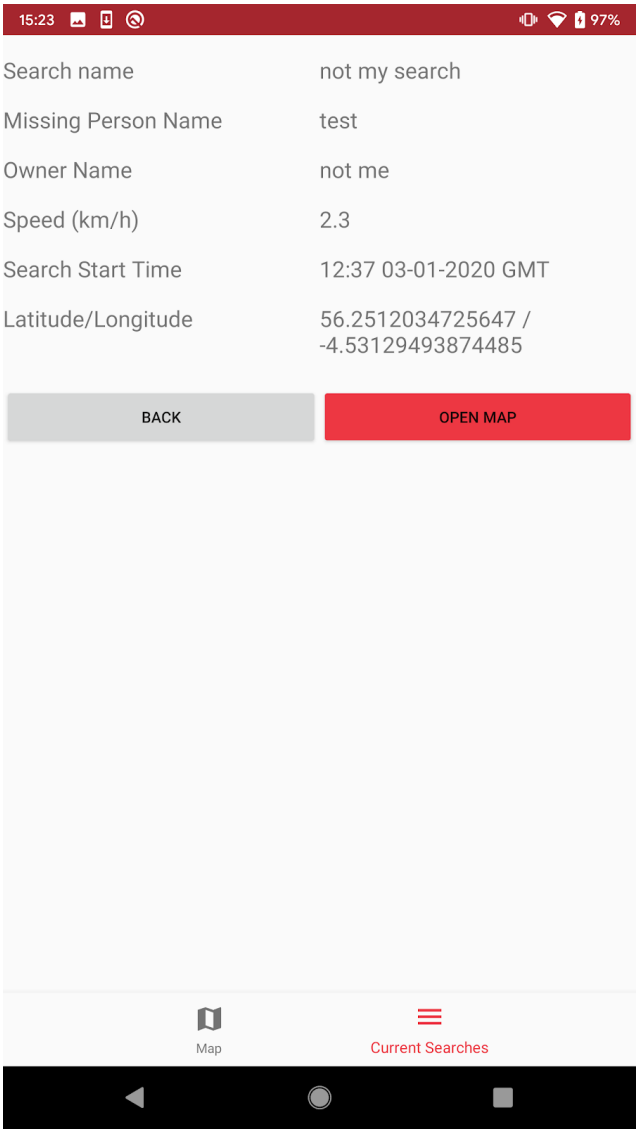


Next I tried to delete the search, and the alert dialog came up the screen, pressing no took me back to the search insight screen (just got rid of the dialog) which is what it is supposed to do. Pressing yes, however, deleted the search successfully from the database.

Lastly, I wanted to test what would happen if the ownerID in the database did not match that of the testing device, it should just show the search and not the delete button.



This is the data that is now stored in the database, it has an ownerID which does not match the one on the testing device so there should be no delete button.



This does as expected, so no modifications need to be made to the function as the button is not there.

Review

What was done?

The ability to delete searches if you are the owner, has been added in this iteration. This allows the owner to end the search if and when they need to.

How was it tested?

It was tested using a search created and viewed on the same device and then a search created on a different device and viewed on the original device to see if the user was able to delete the search if they did or did not own it.

Success Criteria Met

- 5 - Simple Design
- 13 - Deleting Searches

Stakeholder Feedback

Now that the iterative development section of the solution was complete, I showed the stakeholders how to use the app and got their feedback on it.

James: I think the app has turned out quite well. I like how the search area expands as time goes on as it saves me from recreating the search. I found it simple to use and the layout was good. I do think that being able to contact us (as the search owners) is quite important so it would be good to see that feature implemented at some point in the future. Other than this though, I think this tool would have potential in the real world if things were perfected. It saves the owners a lot of time, sharing maps and missing person information etc.

Mr Mapstone: In regard to the output of the search area, this is clearly represented by the red shaded area. Using a Google maps style map is helpful as this is something I am very familiar with using. Input of the required date is easy and made clear by the simple design - as a list in the app. Having the portability of this program in mobile app form makes this even more useful. All of the success criteria that were wanted have been met bar of course the final one, but this is less critical.

Sava: As a stakeholder, I have been keenly looking at this project since its inception. From the beginning I have thought the project was interesting and very worthwhile. When I was first asked about the project, I said that I would like the ability to have multiple isochrones visible on the map at once, but to my knowledge this hasn't been implemented which is slightly disappointing but not disastrous. Another thing I would have liked to see was a desktop web-app or something similar to allow the constabulary's command information centre to have access to all of the searches in one place, allowing them to more easily coordinate and search & rescue efforts; however, this was not implemented – I personally think this was an oversight.

The final user interface looks adequate for use within the constabulary; it is very simple which limits the amount of errors that can be made. When making a new search, however, there are quite a few parameters which are needed. I suspect this would become confusing to a layperson. The buttons are clear and obvious which is key to easy operation of the app.

The map API used isn't my favourite but is still very usable, I would have preferred Google Maps or OpenStreetMap. The isochrone itself doesn't have as many points as I was hoping but regardless it looks good and does the job; and there is an obvious clear button which makes operation of the app better.

In conclusion I think that the project has been a success, it addresses all the key difficulties of the current manual system in a simple and easy to use way. Well done, Ben!

Will: I loved the map, it was great and the searches screen was excellent. I am glad that you implemented the expanding search area function. However it did crash, and that wouldn't be great during a search, but Ben told me that there was something in the map library causing this, which is out of his control.

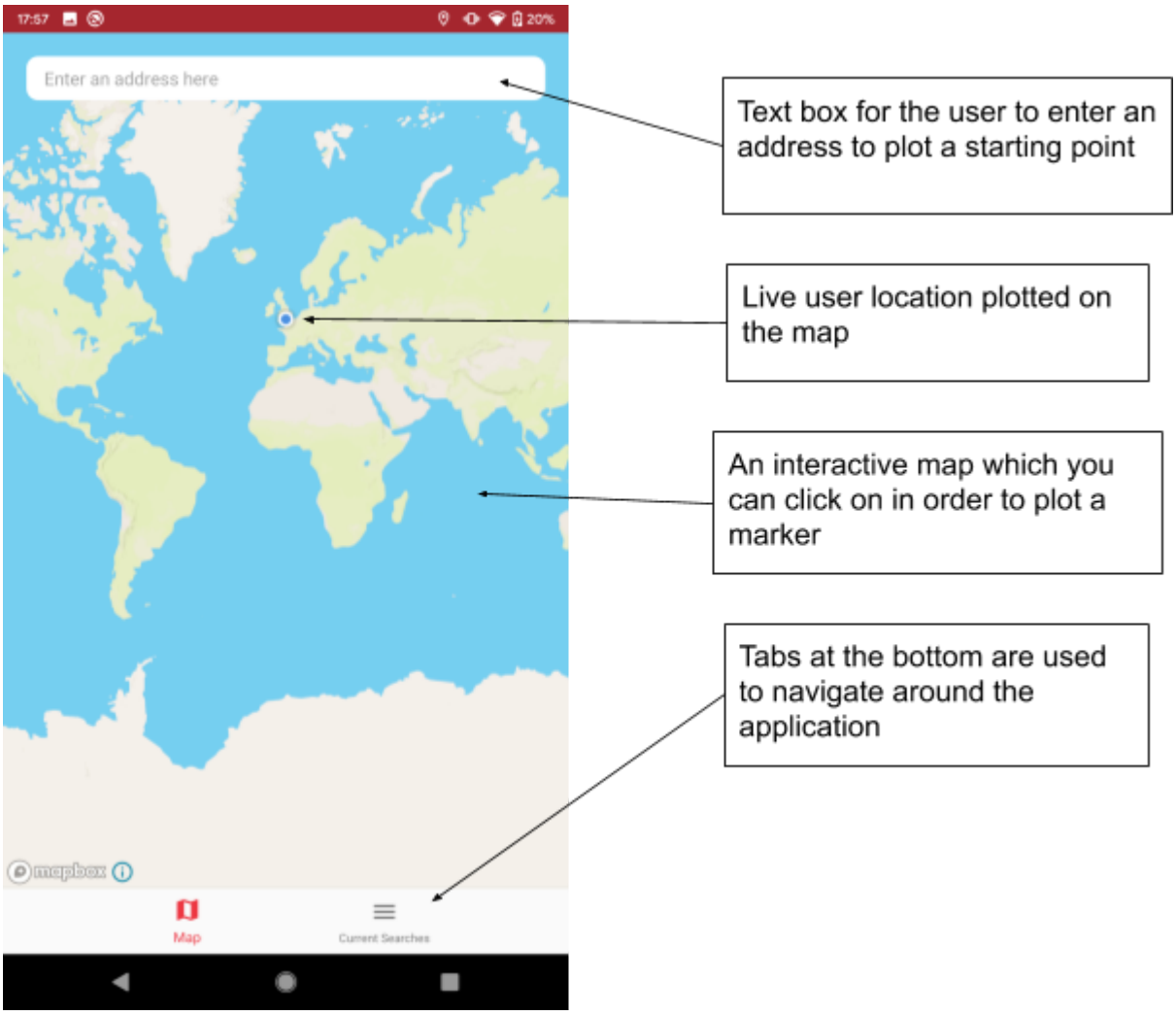
Evaluation

Success Criteria Met

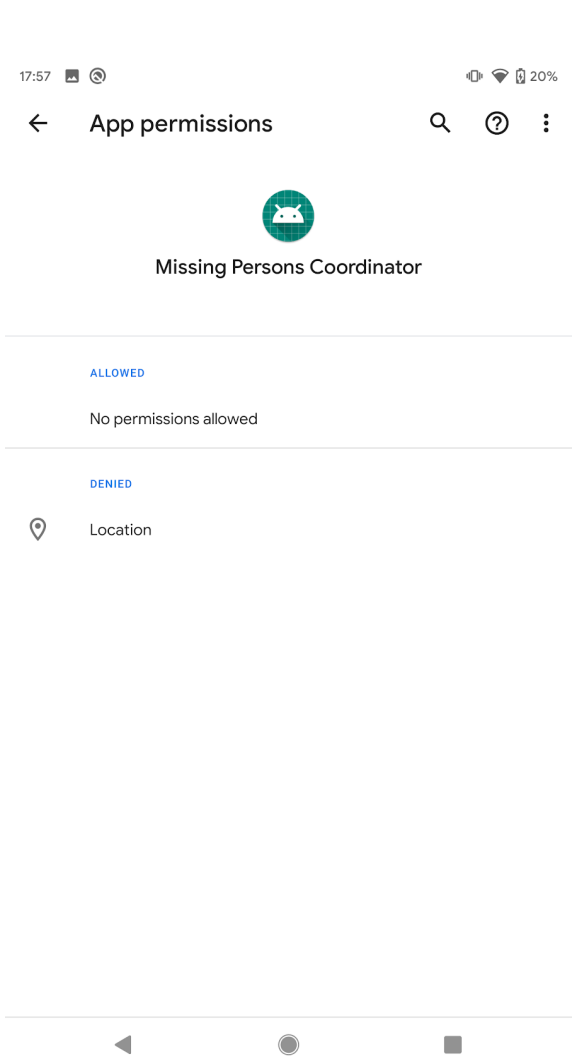
No.	Criteria	Was it met?
1	Interactive map	Yes - Iteration 1
2	User location on map	Yes - Iteration 1
3	Use an address to plot a marker	Yes - Iteration 1
4	Tap the map to plot a marker	Yes - Iteration 1
5	Simple design	Yes - All iterations
6	Search parameter input	Yes - Iteration 2
7	Isochrone algorithm	Yes - Iteration 1 (and iteration 3 with topology)
8	Fetching and applying elevation data	Yes - Iteration 3
9	Display search boundary	Yes - Iteration 1/3
10	Current searches	Yes - Iteration 1
11	Search Insight	Yes - Iteration 2
12	Saving searches to an online database	Yes - Iteration 2
13	Deleting searches	Yes - Iteration 4
14	Contacting the search owner	No

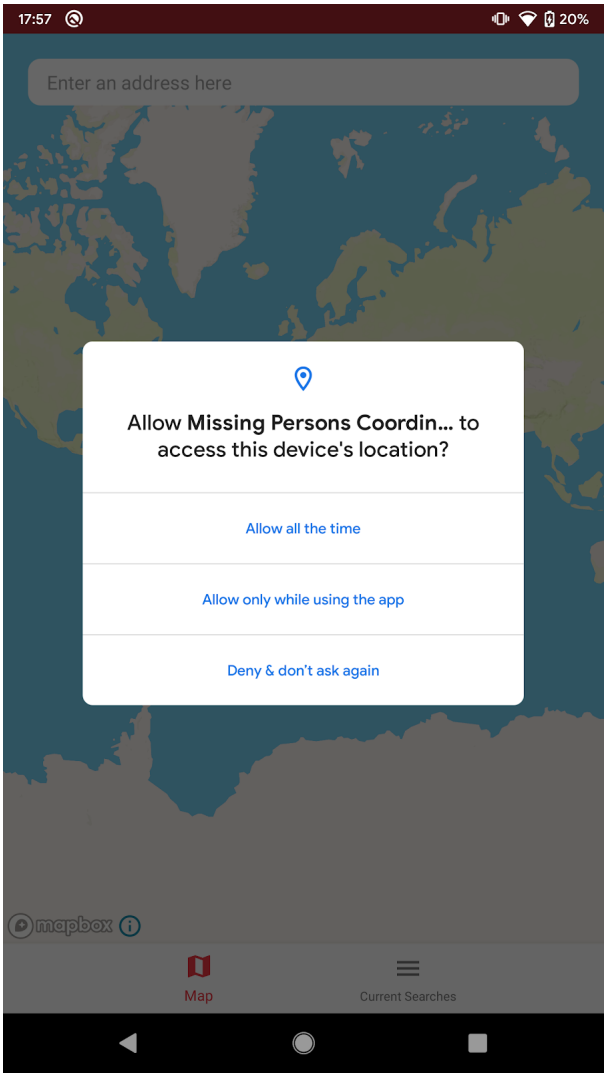
All of the success criteria were met apart from 14, the ability to contact the search owner through the app. I was unable to implement this feature due to time constraints. I also felt that this was not as important as easily being able to share searches with other people as contact fields could just be added to the search insight.

Evidence supporting the success criteria



In this screenshot, criteria 1, 2, 3 and 4 were met. The user is presented with 2 options of adding a marker to the map, either through tapping the map or using the text box at the top which makes a query to an API and then returns and plots the coordinates on a map.
In my code I have also included permission checks, so that if the location gets disabled/denied by the user then the app will request that you allow permission location in order to get maximum functionality. This is shown in the below images, on the left shows me disabling the location permission for the application and then on the right is an example of how the request looks to the user.



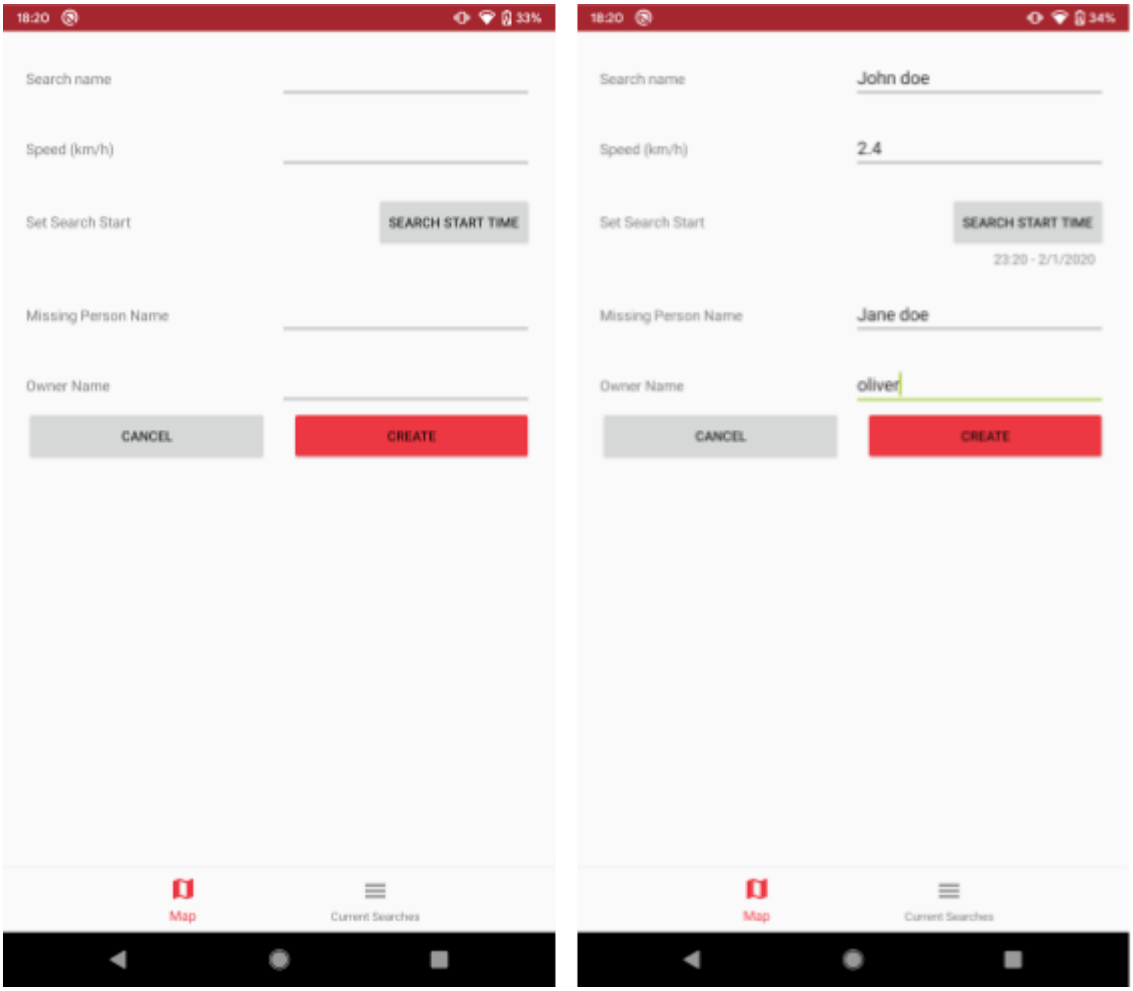


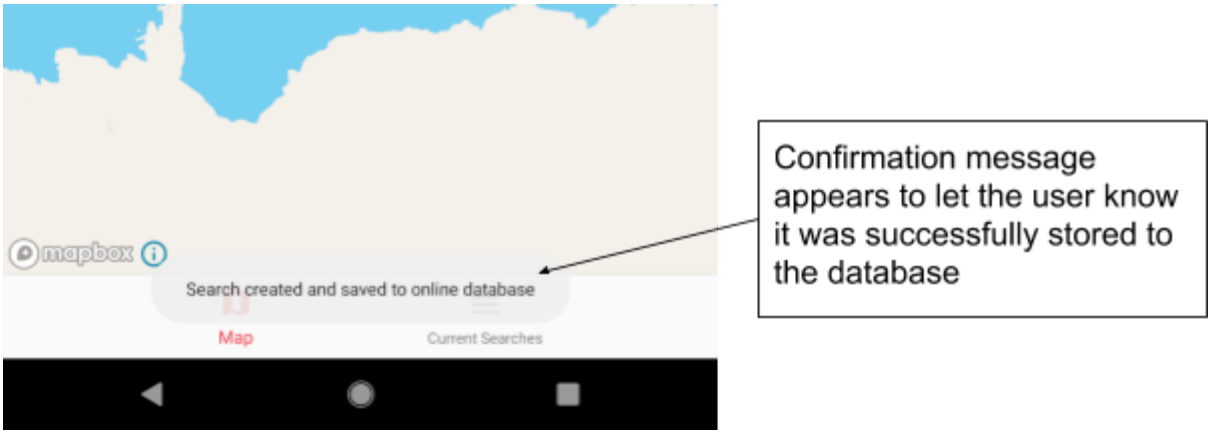
Creating a search:

Search starting point (plotted via tapping)

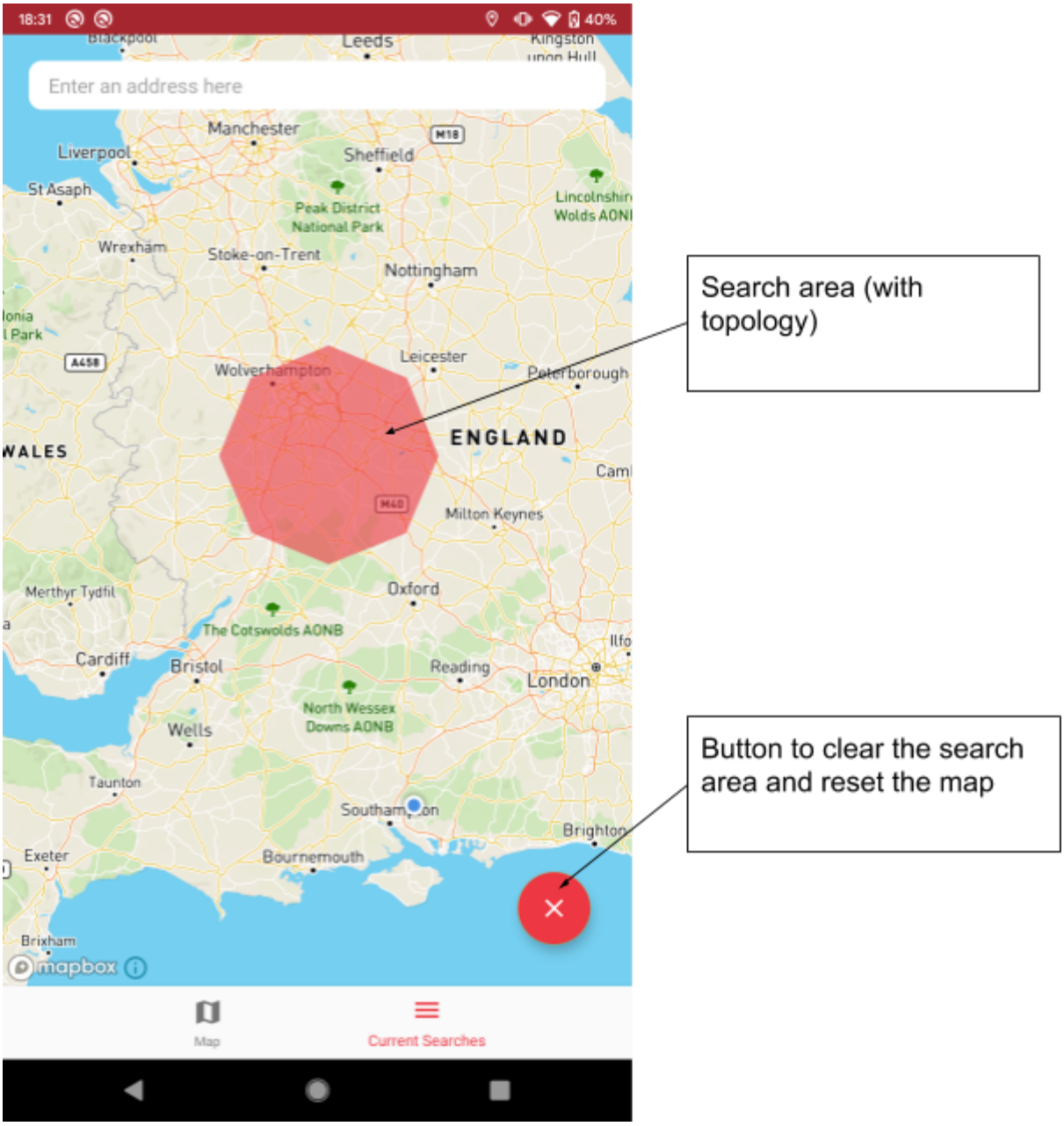
"Start search" button will change the screen to allow the user to input information

"Clear" button will clear the map of any markers and also get rid of the two buttons

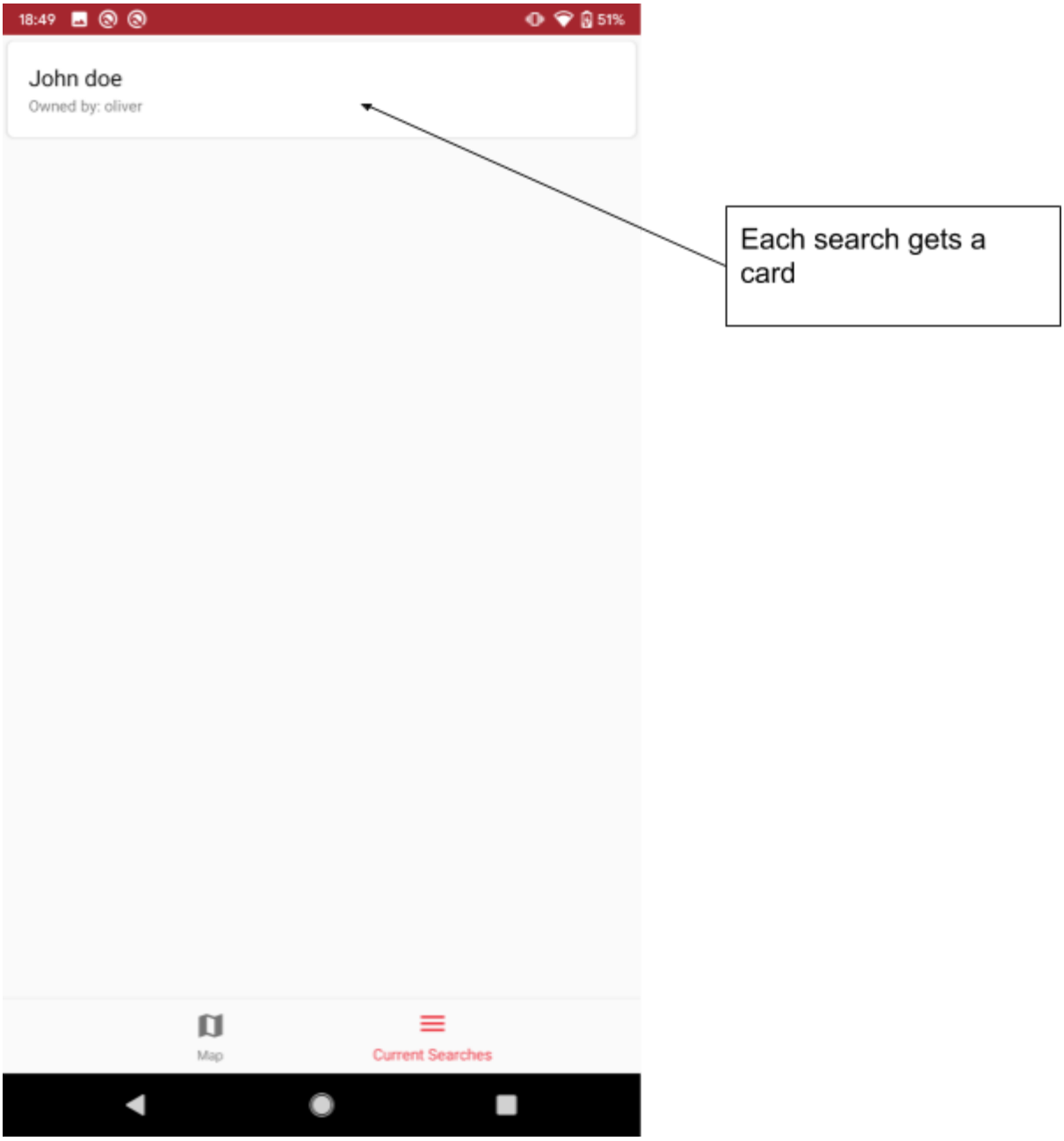


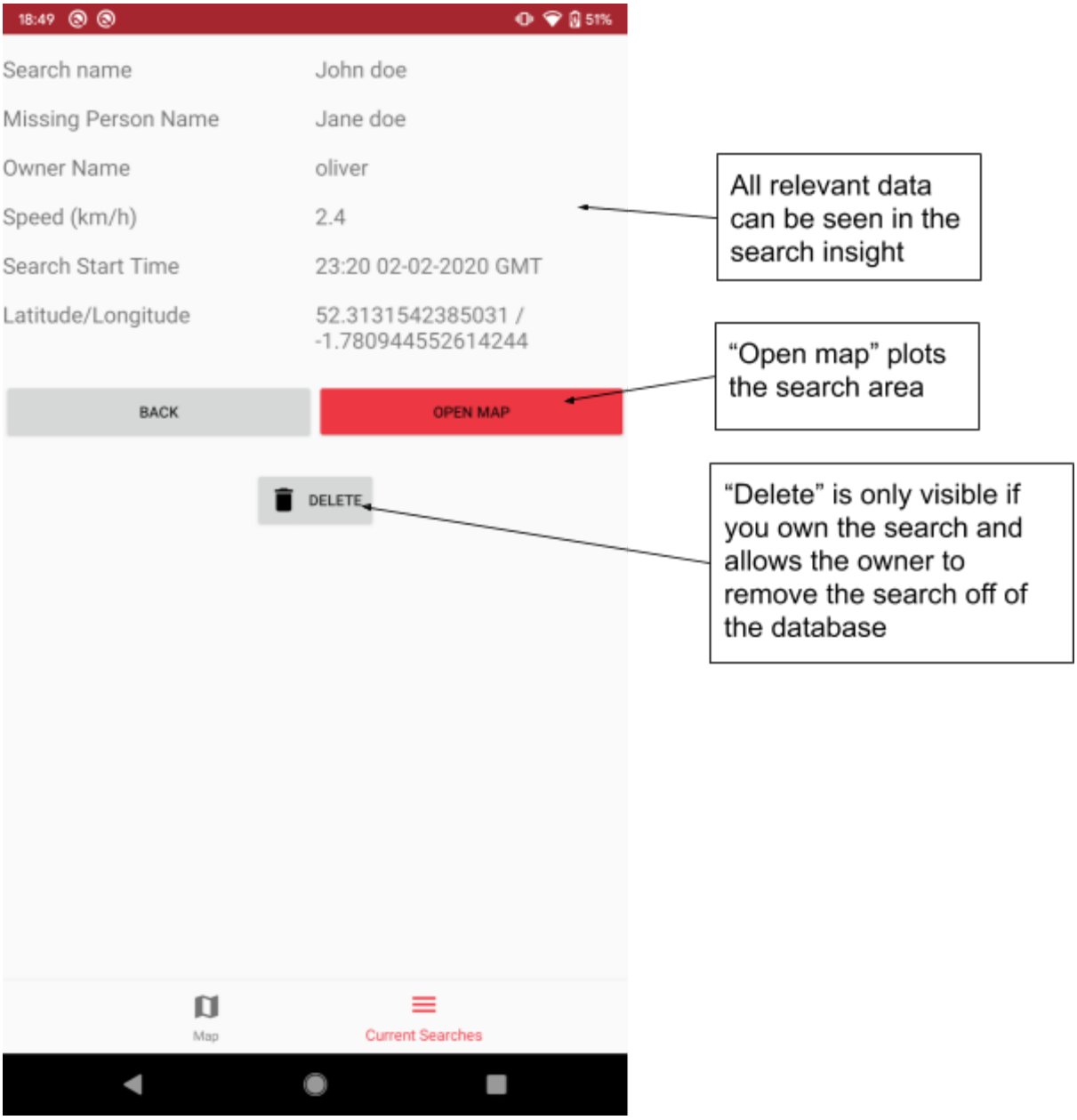


Once the search has been created, topography of the area is collected and applied to the data. The final search area is then plotted on the map.



Here, criteria 6, 7, 8, 9 and 12 are shown and they work as they should. The search boundary shown does have elevation data on it, just the overall height gain/loss was insignificant to make much of an impact to the coordinates.


18:49
51%
John doe
Owned by: oliver
Each search gets a card
Map
Current Searches



This shows testing for criteria 10, 11 and 13 which concludes the project. If I had more time, I would be able to implement the last success criteria as well as other things stakeholders mentioned, such as adding images to search information.

Success criteria 5, a simple design, I tried to meet throughout the project. I think this was completed with success, as all the buttons and inputs have clear labels. The layouts of the screens have been kept fairly minimal so that there is little confusion with what everything does.

Post Development Testing

Tests for text boxes:

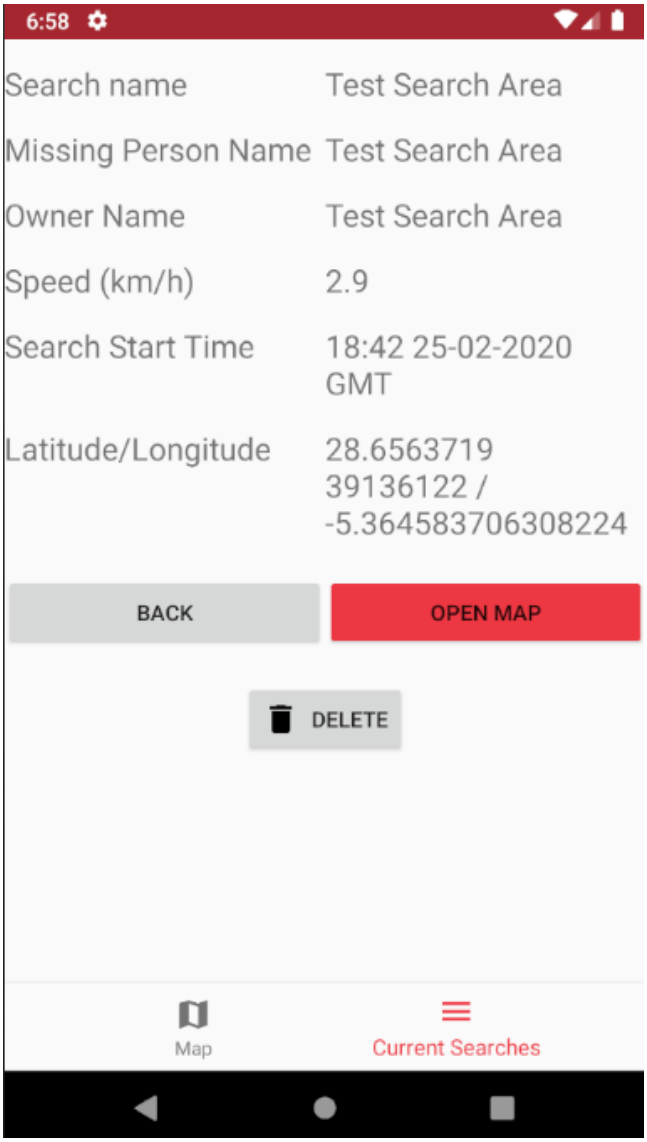
Data Type	Data to test	Expected outcome	Actual outcome
Valid	Test Search Area	Continue as normal	Continue as normal
Valid extreme	Tom/Sarah Missing @ Soton	Continue as normal	Continue as normal
Erroneous	王明失踪 at 12:θθ	Continue as normal	Continue as normal

Once again, all tests carried out with text boxes worked fine as all unicode characters were escaped due to variable assignment in Java. Results show the unicode as it gets reinterpreted by the frontend of the database and the app as well.


```
missingPersonName: "Test Search Area"
owner: "Test Search Area"
ownerID: "6615f010594873eb"
searchName: "Test Search Area"
speed: 2.9000000953674316
startLat: 28.656371939136122
startLon: -5.364583706308224
startTime: 1582656120000

missingPersonName: "Tom/Sarah Missing @ Soton"
owner: "Tom/Sarah Missing @ Soton"
ownerID: "6615f010594873eb"
searchName: "Tom/Sarah Missing @ Soton"
speed: 2.4000000943276
startLat: -69.45014
startLon: 48.35386
startTime: 1582656120000
```

```
missingPersonName: "王明失踪 at 12:00"
owner: "王明失踪 at 12:00"
ownerID: "6615f010594873eb"
searchName: "王明失踪 at 12:00"
speed: 2.1000000948296
startLat: 28.35406
startLon: 20.61927
startTime: 1582656120000
```

Whilst testing this, all buttons were tested and had the expected outcome. The delete button also appeared as I owned all searches in this case.



The testing done after development made sure to check everything in the app, which all turned out fine. Although more extensive tests took place during development because this meant I was able to fix them in the next iteration if there was anything wrong.

Limitations and Potential Improvements

The biggest limitation in my final proposed solution still lies in the accuracy of the search boundary. Whilst my solution does take into account topography of the surrounding land, it still does not give a precise answer, it just aids the process for the police officers. However with more time, I might be able to implement something that would take in information about the missing persons relatives and interests in order to plot areas that they are more likely to be in (such as going staying in a relatives house).

Another limitation is that my solution has only really been designed for those people that are walking. Whilst it would still plot a search area for the speed of a car, it still only takes into account the topography and not data such as traffic.

Being able to share searches with other people was fairly key in my solution, this feature has been completed. Although a limitation was there was no way to contact the owner of the search in order to relay information. I could add a field when creating the search, for the owner to input contact information.

Too improve my solution, I could look into making search boundaries more accurate by calculating more points and then fetching those elevations and applying them too. Another potential improvement is to be able to add more data about the missing person such as an image, or have a custom data table where the owner can add as much data as they'd want to a specific case rather than having to add it into the app. That way they can add extra data about someone when it is necessary rather than making it a compulsory field to fill in on the create search screen.

Maintenance

Android apps, for the most part, are written in Java which is an Object Oriented (OO) language. By using an OO language it makes the code easier to maintain and sub-procedures (or classes) are easily reusable as well as being easy to read and rewrite. I have broken my code down into maintainable classes, the class name describes the role that it plays in the app, for example MapFragment.java will control all of the map logic for the android fragment which is nested inside of an activity (MainActivity.java). This means if someone wanted to update any code to do with the map it is easy to know where to go to edit it.

All code has been commented where necessary although a lot of it is fairly simple to understand whilst reading it as I have given variables sensible names for their purpose. Code has also been indented to make it easier to read.

It could possibly be improved by using JavaDoc which is a Java documentation program that will create a set of docs for the code, this would possibly helps some, but not all developers, depending on how they work best.

If stakeholders were to want new features, these can be easily implemented due to the modular approach of my solution. All that has to be done is another layout file is made with a class to run it and then a button or some form of input on another screen in order to navigate to the new screen. For example, people may want a chat feature to still be implemented so a new fragment could be made with a button on the insight search page and then you are able to send messages directly to the search owner.

Bibliography

https://stackoverflow.com - StackOverflow was quite helpful for diagnosing any problems I had

https://github.com - Github was a useful resource, especially whilst trying to look for the bug I encountered with the Mapbox and live user location

Final Code

In this section, all of the code for this section can be found, only changes will be noted at each iteration and the final code for the whole project can be found at the bottom. Each individual iteration in the Iterative Development section and in the Final Code section do not include the relevant layout files, however they will be in the final code for the whole project.

Iteration 1 - Basic Search Area on Map and Current Searches

ApplicationClass.java

```
package com.bengavin.missingpersons;

import com.mapbox.mapboxsdk.Mapbox;

import androidx.multidex.MultiDexApplication;

public class ApplicationClass extends MultiDexApplication {
```

```
    //Referenced in manifest
    //Initialises map in the onCreate method
    @Override
    public void onCreate() {
        super.onCreate();
        String token = getString(R.string.mapbox_mapsdk_token);
        Mapbox.getInstance(getApplicationContext(), token);
    }

}
```

MainActivity.java

```
package com.bengavin.missingpersons;

import android.os.Bundle;
import android.view.MenuItem;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main); //Show the main activity which will
contain all of the fragments

        BottomNavigationView bottomNav = findViewById(R.id.bottom_navigation);
        bottomNav.setOnNavigationItemSelectedListener(bottomNavListener); //Set a
listener to listen for when the tabs are changed at the bottom nav bar


getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new
MapFragment()).commit(); //Start the app by showing the map fragment
    }

    private BottomNavigationView.OnNavigationItemSelectedListener bottomNavListener
= new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
            Fragment currentFragment = null;
            switch (menuItem.getItemId()) {
                case R.id.nav_map:
                    //Map tab
                    currentFragment = new MapFragment();
                    break;
                case R.id.nav_searches:
                    //Searches tab
                    currentFragment = new SearchesFragment();
```

```
                    break;
            }

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
currentFragment).commit();
            return true;
        }
    };

    protected void onDestroy() {
        super.onDestroy();
    }

}
```

MapFragment.java

```
package com.bengavin.missingpersons;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.Location;
import android.os.Bundle;
import android.os.Looper;
import android.util.Log;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.Toast;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.mapbox.android.core.location.LocationEngine;
import com.mapbox.android.core.location.LocationEngineCallback;
import com.mapbox.android.core.location.LocationEngineProvider;
import com.mapbox.android.core.location.LocationEngineRequest;
import com.mapbox.android.core.location.LocationEngineResult;
import com.mapbox.geojson.Point;
import com.mapbox.geojson.Polygon;
import com.mapbox.mapboxsdk.annotations.PolygonOptions;
import com.mapbox.mapboxsdk.camera.CameraUpdateFactory;
import com.mapbox.mapboxsdk.geometry.LatLng;
import com.mapbox.mapboxsdk.geometry.LatLngBounds;
import com.mapbox.mapboxsdk.location.LocationComponent;
import com.mapbox.mapboxsdk.location.LocationComponentActivationOptions;
import com.mapbox.mapboxsdk.location.modes.CameraMode;
import com.mapbox.mapboxsdk.location.modes.RenderMode;
import com.mapbox.mapboxsdk.maps.MapView;
import com.mapbox.mapboxsdk.maps.MapboxMap;
```

```
import com.mapbox.mapboxsdk.maps.OnMapReadyCallback;
import com.mapbox.mapboxsdk.maps.Style;
import com.mapbox.mapboxsdk.plugins.annotation.Fill;
import com.mapbox.mapboxsdk.plugins.markerview.MarkerView;
import com.mapbox.mapboxsdk.plugins.markerview.MarkerViewManager;
import com.mapbox.mapboxsdk.style.layers.FillLayer;
import com.mapbox.mapboxsdk.style.sources.GeoJsonSource;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.lang.ref.WeakReference;
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.List;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

import static com.mapbox.mapboxsdk.style.layers.PropertyFactory.fillColor;
import static com.mapbox.mapboxsdk.style.layers.PropertyFactory.fillOpacity;

public class MapFragment extends Fragment implements OnMapReadyCallback,
MapboxMap.OnMapClickListener {

   private static final long DEFAULT_INTERVAL = 5000L; //Time interval for location
to be checked
   private static final long DEFAULT_MAX_TIME = DEFAULT_INTERVAL * 5; //Max
interval for location
   private MainActivity activity;
   private MapView mapView;
   private MapboxMap mapboxMap;
   private Context context;
   private LocationEngine locationEngine;
   private MarkerViewManager markerViewManager;
   private MarkerView markerView;
   private double startLat, startLon;
   private LocationChangeListeningActivityLocationCallback callback = new
LocationChangeListeningActivityLocationCallback(this);
   private Bundle startBundle;
   private boolean polygonToPlot = false;

   //Most overridden methods come from the android fragment lifecycle which can be
found here: https://developer.android.com/guide/components/fragments

   @Override
   public void onAttach(Context context) {
```

```
        super.onAttach(context);
        Context activity = context;
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
        //When the fragment gets created this will inflate the fragment view
        View fragmentView = inflater.inflate(R.layout.fragment_map, container,
false);
        return fragmentView;
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        activity = (MainActivity)context;
        super.onViewCreated(view, savedInstanceState);
        mapView = (MapView) view.findViewById(R.id.mapView);
        mapView.onCreate(savedInstanceState);
        mapView.getMapAsync(this);

        startBundle = this.getArguments();
        if(startBundle != null) {
            polygonToPlot = startBundle.getBoolean("polygonToPlot", false);
        }

        final EditText mapSearchBox = (EditText)
view.findViewById(R.id.map_search_bar);
        mapSearchBox.setOnKeyListener(new View.OnKeyListener() { //Listen for key
presses from the search box at the top, if it is the "enter" key, take value from
the EditText and call another method
            @Override
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                if((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode ==
KeyEvent.KEYCODE_ENTER)) {
                    addressToCoordinates(mapSearchBox.getText().toString()); //Run
addressToCoordinates with value from EditText as string
                }
                return false;
            }
        });

        Button startSearch = getActivity().findViewById(R.id.startSearchButton);
        startSearch.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showSearchInput();
            }
        });

        Button clearScreen = getActivity().findViewById(R.id.clearButton);
        clearScreen.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                clearScreen();
            }
        });
    }

    @Override
    public void onMapReady(@NonNull MapboxMap mapboxMap) {
        this.mapboxMap = mapboxMap;
        mapboxMap.setStyle(Style.MAPBOX_STREETS, new Style.OnStyleLoaded() {
            @Override
            public void onStyleLoaded(@NonNull Style style) {
                markerViewManager = new MarkerViewManager(mapView, mapboxMap);
                enableLocation(style);
                if(polygonToPlot) {
                    float speedInputValue = startBundle.getFloat("speedInputValue",
3);
                    long startTime = startBundle.getLong("startTime",
System.currentTimeMillis());
                    double lat = startBundle.getDouble("lat", 0);
                    double lon = startBundle.getDouble("lon", 0);
                    List<LatLng> points = createIsochrone(speedInputValue,
startTime);
                    Isochrone isochrone = new Isochrone();
                    long time = isochrone.getTimeDifference(startTime);
                    double distance = isochrone.getDistanceTravelled(time,
speedInputValue);
                    double[] pair = {lat, lon};
                    ArrayList<Double> coordinates =
isochrone.getMaxCoordinates(pair, distance);
                    List<List<Point>> POINTS = new ArrayList<>();
                    List<Point> OUTER_POINTS = new ArrayList<>();
                    OUTER_POINTS.add(Point.fromLngLat(coordinates.get(1),
coordinates.get(0)));
                    OUTER_POINTS.add(Point.fromLngLat(coordinates.get(3),
coordinates.get(2)));
                    OUTER_POINTS.add(Point.fromLngLat(coordinates.get(5),
coordinates.get(4)));
                    OUTER_POINTS.add(Point.fromLngLat(coordinates.get(7),
coordinates.get(6)));
                    OUTER_POINTS.add(Point.fromLngLat(coordinates.get(9),
coordinates.get(8)));
                    OUTER_POINTS.add(Point.fromLngLat(coordinates.get(11),
coordinates.get(10)));
                    OUTER_POINTS.add(Point.fromLngLat(coordinates.get(13),
coordinates.get(12)));
                    OUTER_POINTS.add(Point.fromLngLat(coordinates.get(15),
coordinates.get(14)));
                    POINTS.add(OUTER_POINTS);
                    style.addSource(new GeoJsonSource("34",
Polygon.fromLngLats(POINTS)));
                    style.addLayerBelow(new FillLayer("12", "34").withProperties(
                            fillColor(Color.parseColor("#ED3742")),
fillOpacity(0.4f)), "style"
```

```
                    );
                    LatLngBounds latLngBounds = new LatLngBounds.Builder()
                            .include(new LatLng(coordinates.get(0),
coordinates.get(1)))
                            .include(new LatLng(coordinates.get(2),
coordinates.get(3)))
                            .include(new LatLng(coordinates.get(4),
coordinates.get(5)))
                            .include(new LatLng(coordinates.get(6),
coordinates.get(7)))
                            .include(new LatLng(coordinates.get(8),
coordinates.get(9)))
                            .include(new LatLng(coordinates.get(10),
coordinates.get(11)))
                            .include(new LatLng(coordinates.get(12),
coordinates.get(13)))
                            .include(new LatLng(coordinates.get(14),
coordinates.get(15)))
                            .build();

mapboxMap.animateCamera(CameraUpdateFactory.newLatLngBounds(latLngBounds, 50),
3000);
                    FloatingActionButton floatingActionButton =
mapView.findViewById(R.id.clearFloatingActionButton);
                    floatingActionButton.show();
                    floatingActionButton.setOnClickListener(new
View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            MapFragment mapFragment = new MapFragment();

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
                        }
                    });
                }
            }
        });
        mapboxMap.addOnMapClickListener(this);
    }

    @Override
    public boolean onMapClick(@NonNull LatLng point) {
        if(markerView != null) {
            markerViewManager.removeMarker(markerView);
            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            markerView = new MarkerView(point, customView);
            markerViewManager.addMarker(markerView);
            setStartLat(point.getLatitude());
            setStartLon(point.getLongitude());
```

```
            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
            view.setVisibility(View.VISIBLE);
        } else {
            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            markerView = new MarkerView(point, customView);
            markerViewManager.addMarker(markerView);
            setStartLat(point.getLatitude());
            setStartLon(point.getLongitude());
            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
            view.setVisibility(View.VISIBLE);
        }
        return true;
    }

    public List<LatLng> createIsochrone(float speed, long startTime) {
        List<LatLng> OUTER_POINTS = new ArrayList<>();
        // Start point comes from startPoint which gets updated no matter which
method of choosing a start point is used
        Isochrone isochrone = new Isochrone();
        long time = isochrone.getTimeDifference(startTime);
        double distance = isochrone.getDistanceTravelled(time, speed);
        double[] pair = {startLat, startLon};
        ArrayList<Double> coordinates = isochrone.getMaxCoordinates(pair, distance);
        OUTER_POINTS.add(new LatLng(coordinates.get(0), coordinates.get(1)));
        OUTER_POINTS.add(new LatLng(coordinates.get(2), coordinates.get(3)));
        OUTER_POINTS.add(new LatLng(coordinates.get(4), coordinates.get(5)));
        OUTER_POINTS.add(new LatLng(coordinates.get(6), coordinates.get(7)));
        OUTER_POINTS.add(new LatLng(coordinates.get(8), coordinates.get(9)));
        OUTER_POINTS.add(new LatLng(coordinates.get(10), coordinates.get(11)));
        OUTER_POINTS.add(new LatLng(coordinates.get(12), coordinates.get(13)));
        OUTER_POINTS.add(new LatLng(coordinates.get(14), coordinates.get(15)));
        return OUTER_POINTS;
    }

    @SuppressWarnings({"MissingPermission"})
    private void enableLocation(@NonNull Style mapStyle) {
        if(ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
            activity = (MainActivity) getContext(); //Essentially replacing "this"
as "this" can't be used in fragments as there is no context passed here
            LocationComponent locationComponent = mapboxMap.getLocationComponent();
            LocationComponentActivationOptions locationComponentActivationOptions =
LocationComponentActivationOptions.builder(activity,
mapStyle).useDefaultLocationEngine(false).build();

locationComponent.activateLocationComponent(locationComponentActivationOptions);
            locationComponent.setLocationComponentEnabled(true);
            locationComponent.setCameraMode(CameraMode.TRACKING);
```

```
            //locationComponent.setRenderMode(RenderMode.COMPASS); Removed due to a
known issue with the library here:
https://github.com/mapbox/mapbox-gl-native/issues/14889 - Reported fixed
https://github.com/mapbox/mapbox-gl-native-android/pull/19 with build 8.5.0-beta.1
but error still occurring so it has been commented out

            initLocationEngine();
        } else {
            requestPermissions(
                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    1
            ); //Request location permission if not already granted
        }
    }

    @SuppressWarnings({"MissingPermission"})
    private void initLocationEngine() {
        locationEngine = LocationEngineProvider.getBestLocationEngine(activity);
//Initialise new location engine
        LocationEngineRequest request = new
LocationEngineRequest.Builder(DEFAULT_INTERVAL)
                .setPriority(LocationEngineRequest.PRIORITY_HIGH_ACCURACY)
                .setMaxWaitTime(DEFAULT_MAX_TIME).build(); //Set time intervals for
updating user location (2s)
        locationEngine.requestLocationUpdates(request, callback,
Looper.getMainLooper());
        locationEngine.getLastLocation(callback); //Pass it to callback (ie
LocationChangeListeningActivityLocationCallback)
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        //Method called when requestPermissions(); gets called
        if(requestCode == 1) {
            if (permissions[0].equals(Manifest.permission.ACCESS_FINE_LOCATION) &&
grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                //Check if location permissions have been granted
                mapboxMap.getStyle(new Style.OnStyleLoaded() {
                    @Override
                    public void onStyleLoaded(@NonNull Style style) {
                        enableLocation(style); //Rerun method, this time with
correct permissions
                    }
                });
            }
        } else {
            Toast.makeText(getActivity(), R.string.location_access_not_permitted,
Toast.LENGTH_LONG).show(); //Let user know they need to allow location permission
        }
    }

    private void addressToCoordinates(@NonNull String address) {
        address = address.replaceAll("/[^a-zA-Z0-9 ]/g", "").replaceAll(" ", "%20");
```

```
        String url = "https://api.mapbox.com/geocoding/v5/mapbox.places/" + address
+ ".json?&access_token=" + getString(R.string.mapbox_token);
        //Variables must be declared before starting a thread as they have to be
final or effectively-final
        //Creating a thread - running network operations must happen on another
thread as the main/UI thread can skip frames or lifecycle methods
        Thread thread = new Thread(() -> {
            Thread.currentThread().setPriority(Thread.MIN_PRIORITY); //Important to
set the thread priority to MIN or at least less than main/UI thread to prevent
skipping frames or lifecycle methods
            OkHttpClient okHttpClient = new OkHttpClient(); //Initialising an
instance of the HTTP client
            Request request = new Request.Builder() //Build request with URL and
optional headers (not needed)
                    .url(url)
                    .build();
            Response response = null; //Cannot run query in a try(query here){} due
to a target API mismatch (so it runs on more devices)
            try {
                response = okHttpClient.newCall(request).execute();
                String responseStr = response.body().string();
                JSONObject jsonObject = new JSONObject(responseStr); //Convert the
string into a JSONObject for manipulation and data reading
                JSONArray allFeatures = jsonObject.getJSONArray("features");
                JSONObject firstObject = allFeatures.getJSONObject(0);
                JSONObject geometryObject = firstObject.getJSONObject("geometry");
                JSONArray coordPair = geometryObject.getJSONArray("coordinates");
//Final coordinate pair taken from API response in order to plot marker on the map
                double lat = ((Number) coordPair.get(1)).doubleValue();
                double lng = ((Number) coordPair.get(0)).doubleValue();
                setStartLat(lat);
                setStartLon(lng);
                getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if(markerView != null) {
                            markerViewManager.removeMarker(markerView);
                            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
                            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                            markerView = new MarkerView(new LatLng(lat, lng),
customView);
                            markerViewManager.addMarker(markerView);
                            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
                            view.setVisibility(View.VISIBLE);
                        } else {
                            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
                            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
```

```
                            markerView = new MarkerView(new LatLng(lat, lng),
customView);
                            markerViewManager.addMarker(markerView);
                            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
                            view.setVisibility(View.VISIBLE);
                        }
                    }
                });
            } catch (JSONException e) {
                Toast.makeText(getActivity(), "Failed to plot point\nPlease try
again later", Toast.LENGTH_LONG).show();
            } catch(IOException e) {
                Toast.makeText(getActivity(), R.string.ioexception_error_message,
Toast.LENGTH_LONG).show();
            } finally {
                if(response != null) {
                    response.body().close(); //Close the client after checking it
existed in the first place
                }
            }
        });
        thread.start();
    }

    private void clearScreen() {
        View view = (View) getActivity().findViewById(R.id.startSearchRectangle);
        if (view.getVisibility()== View.VISIBLE) {
            view.setVisibility(View.GONE);
        }
        markerViewManager.removeMarker(markerView);
        final EditText mapSearchBox = (EditText)
getActivity().findViewById(R.id.map_search_bar);
        mapSearchBox.getText().clear();
    }

    private void showSearchInput() {
        View view = (View) getActivity().findViewById(R.id.startSearchRectangle);
        if (view.getVisibility() == View.VISIBLE) {
            view.setVisibility(View.GONE);
        }
        Fragment newSearchFragment = new NewSearchFragment();
        Bundle bundle = new Bundle();
        bundle.putDouble("lat", startLat);
        bundle.putDouble("lon", startLon);
        newSearchFragment.setArguments(bundle);
        getFragmentManager().beginTransaction().replace(R.id.fragment_container,
newSearchFragment).commit();
    }

    private static class LocationChangeListeningActivityLocationCallback implements
LocationEngineCallback<LocationEngineResult> {

        private final WeakReference<MapFragment> fragmentWeakReference;
```

```
        LocationChangeListeningActivityLocationCallback(MapFragment fragment) {
            this.fragmentWeakReference = new WeakReference<>(fragment);
        }

        @Override
        public void onSuccess(LocationEngineResult result) {
            MapFragment fragment = fragmentWeakReference.get();

            if(fragment != null) {
                Location location = result.getLastLocation(); //Fetches last
location

                if(location == null) {
                    return;
                }
                if(fragment.mapboxMap != null && result.getLastLocation() != null) {

fragment.mapboxMap.getLocationComponent().forceLocationUpdate(result.getLastLocatio
n()); //Forces map to update user location to last lat lon from location engine
                }
            }
        }

        @Override
        public void onFailure(@NonNull Exception ignored) {

        }

    }

    private double getStartLat() {
        return this.startLat;
    }

    private void setStartLat(double lat) {
        this.startLat = lat;
    }

    private double getStartLon() {
        return this.startLon;
    }

    private void setStartLon(double lon) {
        this.startLon = lon;
    }

    @SuppressWarnings({"MissingPermission"})
    @Override
    public void onStart() {
        super.onStart();
        mapView.onStart();
    }
```

```
    @Override
    public void onSaveInstanceState(@NonNull Bundle outState) {
        super.onSaveInstanceState(outState);
        mapView.onSaveInstanceState(outState);
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        if(locationEngine != null) {
            locationEngine.removeLocationUpdates(callback);
        }
        if(mapView != null) {
            mapView.onDestroy();
        }
    }

    @Override
    public void onPause() {
        super.onPause();
        mapView.onPause();
    }

    @Override
    public void onResume() {
        super.onResume();
        mapView.onResume();
    }

}
```

SearchesFragment.java

```
package com.bengavin.missingpersons;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
```

```
public class SearchesFragment extends Fragment {

   private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
   RecyclerView recyclerView;
   CardAdapter cardAdapter;

   @Nullable
   @Override
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       View fragmentView = inflater.inflate(R.layout.fragment_searches, container,
false);
       return fragmentView;
   }

   @Override
   public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
       super.onViewCreated(view, savedInstanceState);

       recyclerView = view.findViewById(R.id.recyclerView);
       recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

       getSearches();
   }

   private void getSearches() {
       onlineDatabase.collection("searches")
               .get()
               .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                   @Override
                   public void onComplete(@NonNull Task<QuerySnapshot> task) {
                       ArrayList<CardModel> cards = new ArrayList<>();
                       if(task.isSuccessful()) {
                           for(QueryDocumentSnapshot documentSnapshot :
task.getResult()) {
                               CardModel cardModel = new CardModel();

cardModel.setSearchName(documentSnapshot.getString("searchName"));
                               cardModel.setSearchOwner("Owned by: " +
documentSnapshot.getString("owner"));
                               cardModel.setUid(documentSnapshot.getId());
                               cards.add(cardModel);
                           }
                           cardAdapter = new CardAdapter(getActivity(), cards);
                           recyclerView.setAdapter(cardAdapter);
                       } else {
                           Toast.makeText(getContext(), "Failed to load
searches\nPlease try again", Toast.LENGTH_LONG).show();
                       }
                   }
               });
   }

}
```

CardAdapter.java

```
package com.bengavin.missingpersons;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.recyclerview.widget.RecyclerView;

public class CardAdapter extends RecyclerView.Adapter<CardHolder> {

   Context context;
   ArrayList<CardModel> cardModels;

   public CardAdapter(Context context, ArrayList<CardModel> cardModels) {
       this.context = context;
       this.cardModels = cardModels;
   }

   @NonNull
   @Override
   public CardHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
       View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.search_card, null);
//Inflate search_card.xml when the view holder gets created

       return new CardHolder(view);
   }

   @Override
   public void onBindViewHolder(@NonNull CardHolder holder, int position) {
       holder.mSearchName.setText(cardModels.get(position).getSearchName());
       holder.mSearchOwner.setText(cardModels.get(position).getSearchOwner());
   }

   @Override
   public int getItemCount() {
       return cardModels.size();
   }
}
```

CardHolder.java

```
package com.bengavin.missingpersons;

import android.view.View;
```

```
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class CardHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

   TextView mSearchName, mSearchOwner;
   CardClickListener cardClickListener;

   CardHolder(@NonNull View itemView) {
       super(itemView);

       this.mSearchName = itemView.findViewById(R.id.searchName);
       this.mSearchOwner = itemView.findViewById(R.id.searchOwner);
   }

}
```

CardModel.java

```
package com.bengavin.missingpersons;

public class CardModel {

   //This class just contains getters and setters for the card model

   private String searchName, searchOwner, uid;

   public String getSearchName() {
       return searchName;
   }

   public void setSearchName(String searchName) {
       this.searchName = searchName;
   }

   public String getSearchOwner() {
       return searchOwner;
   }

   public void setSearchOwner(String searchOwner) {
       this.searchOwner = searchOwner;
   }

   public String getUid() {
       return uid;
   }

   public void setUid(String uid) {
       this.uid = uid;
   }

}
```

Isochrone.java

```
package com.bengavin.missingpersons;

import android.util.Log;

import java.lang.reflect.Array;
import java.util.ArrayList;

public class Isochrone {

   private double toRadians(double degrees) {
       double radians = degrees * ((Math.PI)/180);
       return radians;
   }

   private double toDegrees(double radians) {
       double factor = 180 / Math.PI;
       return radians * factor;
   }

   public ArrayList<Double> getMaxCoordinates(double[] startingPair, double
distance) {
       double lat = toRadians(startingPair[0]);
       double lon = toRadians(startingPair[1]);
       double distanceRatio = distance/6371.01; //Divide the distance given by the
radius of the Earth to get a ratio
       ArrayList<Double> endMaxCoordinates = new ArrayList<Double>(); //Initialise
end array for all coordinates (size 16 as 8 pairs)
       for(int i = 0; i < 360; i+= 45) { // <360 as 360 should not be included as
it is the same as 0
           double bearing = toRadians(i);
           double endLat = Math.asin(Math.sin(lat)*Math.cos(distanceRatio) +
Math.cos(lat)*Math.sin(distanceRatio)*Math.cos(bearing));
           endMaxCoordinates.add(toDegrees(endLat));
           double endLon = lon +
Math.atan2(Math.sin(bearing)*Math.sin(distanceRatio)*Math.cos(lat),
Math.cos(distanceRatio)-Math.sin(lat)*Math.sin(endLat));
           endMaxCoordinates.add(toDegrees(endLon));
       }
       return endMaxCoordinates;
   }

   public long getTimeDifference(long startTime) {
       startTime = startTime/1000L;
       long unixTime = System.currentTimeMillis()/1000L;
       long difference = unixTime - startTime;
       return difference;
   }

   public double getDistanceTravelled(double timeDifference, double speed) {
       timeDifference = timeDifference/60/60;  // Gets time in hours
       double displacement = timeDifference * speed;
       return displacement; // Displace in km/h
```

```
    }

}
```

NewSearchFragment.java

```
package com.bengavin.missingpersons;

import android.app.AlertDialog;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.mapbox.mapboxsdk.geometry.LatLng;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class NewSearchFragment extends Fragment {

    private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
    private Button cancelButton, submitButton, setDateTimeButton;
    private EditText searchName, speedInput, missingPersonName, ownerName;
    private TextView timeOutput;
    private Bundle bundle;
    private long time = 0;
    private double lat, lon;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
        View fragmentView = inflater.inflate(R.layout.fragment_new_search,
container, false);
        return fragmentView;
    }
```

```
    @Override
    public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        setDateTimeButton = view.findViewById(R.id.set_date_time);
        cancelButton = view.findViewById(R.id.cancel_button);
        submitButton = view.findViewById(R.id.submit_button);
        timeOutput = view.findViewById(R.id.time_picker_output);

        bundle = this.getArguments();
        lat = bundle.getDouble("lat", 0);
        lon = bundle.getDouble("lon", 0);

        setDateTimeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final View dialogView = View.inflate(getActivity(),
R.layout.date_time_picker, null);
                final AlertDialog alertDialog = new
AlertDialog.Builder(getActivity()).create();

                dialogView.findViewById(R.id.set).setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View inner) {
                        DatePicker datePicker =
alertDialog.findViewById(R.id.date_picker);
                        TimePicker timePicker =
alertDialog.findViewById(R.id.time_picker);

                        timePicker.setIs24HourView(true);

                        Calendar calendar = new
GregorianCalendar(datePicker.getYear(), datePicker.getMonth(),
datePicker.getDayOfMonth(), timePicker.getCurrentHour(),
timePicker.getCurrentMinute());

                        time = calendar.getTimeInMillis();

timeOutput.setText(getString(R.string.date_time_output_template,
String.valueOf(timePicker.getCurrentHour()),
String.valueOf(timePicker.getCurrentMinute()),
String.valueOf(datePicker.getDayOfMonth()), String.valueOf(datePicker.getMonth()),
String.valueOf(datePicker.getYear())));
                        alertDialog.dismiss();
                    }
                });
                alertDialog.setView(dialogView);
                alertDialog.show();
            }
        });

        cancelButton.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {

getFragmentManager().beginTransaction().replace(R.id.fragment_container, new
MapFragment()).commit();
            }
        });

        submitButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                searchName = view.findViewById(R.id.search_name_input);
                String searchNameInput = searchName.getText().toString();
                speedInput = view.findViewById(R.id.speed_input);
                missingPersonName =
view.findViewById(R.id.missing_person_name_input);
                String missingPersonNameInput =
missingPersonName.getText().toString();
                ownerName = view.findViewById(R.id.search_owner_input);
                String ownerNameInput = ownerName.getText().toString();
                float speedInputValue = 0;
                try {
                    speedInputValue =
Float.parseFloat(speedInput.getText().toString());
                    if(searchNameInput.length() == 0 ||
missingPersonNameInput.length() == 0 || ownerNameInput.length() == 0) {
                        Toast.makeText(getActivity(), "Please fill in all of the
fields", Toast.LENGTH_SHORT).show();
                        if (time == 0) {
                            time = System.currentTimeMillis();
                        }
                    } else {
                        createSearch(searchNameInput, ownerNameInput,
bundle.getDouble("lat", 0), bundle.getDouble("lon", 0), speedInputValue,
missingPersonNameInput, time);
                    }
                } catch (NumberFormatException e) {
                    Toast.makeText(getActivity(), "Please enter a speed",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private void createSearch(String searchName, String ownerName, double startLat,
double startLon, float speed, String missingPersonName, long time) {

        Toast.makeText(getActivity(), "Search created", Toast.LENGTH_SHORT).show();
        MapFragment mapFragment = new MapFragment();
        Bundle newBundle = new Bundle();
        newBundle.putBoolean("polygonToPlot", true);
        newBundle.putFloat("speedInputValue", speed);
        newBundle.putLong("startTime", time);
        newBundle.putDouble("lat", lat);
```

```
        newBundle.putDouble("lon", lon);
        mapFragment.setArguments(newBundle);
        getFragmentManager().beginTransaction().replace(R.id.fragment_container,
        mapFragment).commit();


    }

}
```

Iteration 2 - Writing new search to database and search insights

NewSearchFragment.java

```
package com.bengavin.missingpersons;

import android.app.AlertDialog;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.mapbox.mapboxsdk.geometry.LatLng;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class NewSearchFragment extends Fragment {

    private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
    private Button cancelButton, submitButton, setDateTimeButton;
    private EditText searchName, speedInput, missingPersonName, ownerName;
```

```
    private TextView timeOutput;
    private Bundle bundle;
    private long time = 0;
    private double lat, lon;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
        View fragmentView = inflater.inflate(R.layout.fragment_new_search,
container, false);
        return fragmentView;
    }

    @Override
    public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        setDateTimeButton = view.findViewById(R.id.set_date_time);
        cancelButton = view.findViewById(R.id.cancel_button);
        submitButton = view.findViewById(R.id.submit_button);
        timeOutput = view.findViewById(R.id.time_picker_output);

        bundle = this.getArguments();
        lat = bundle.getDouble("lat", 0);
        lon = bundle.getDouble("lon", 0);

        setDateTimeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final View dialogView = View.inflate(getActivity(),
R.layout.date_time_picker, null);
                final AlertDialog alertDialog = new
AlertDialog.Builder(getActivity()).create();

                dialogView.findViewById(R.id.set).setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View inner) {
                        DatePicker datePicker =
alertDialog.findViewById(R.id.date_picker);
                        TimePicker timePicker =
alertDialog.findViewById(R.id.time_picker);

                        timePicker.setIs24HourView(true);

                        Calendar calendar = new
GregorianCalendar(datePicker.getYear(), datePicker.getMonth(),
datePicker.getDayOfMonth(), timePicker.getCurrentHour(),
timePicker.getCurrentMinute());

                        time = calendar.getTimeInMillis();

timeOutput.setText(getString(R.string.date_time_output_template,
```

```
String.valueOf(timePicker.getCurrentHour()),
String.valueOf(timePicker.getCurrentMinute()),
String.valueOf(datePicker.getDayOfMonth()), String.valueOf(datePicker.getMonth()),
String.valueOf(datePicker.getYear())));
                        alertDialog.dismiss();
                    }
                });
                alertDialog.setView(dialogView);
                alertDialog.show();
            }
        });

        cancelButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

getFragmentManager().beginTransaction().replace(R.id.fragment_container, new
MapFragment()).commit();
            }
        });

        submitButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                searchName = view.findViewById(R.id.search_name_input);
                String searchNameInput = searchName.getText().toString();
                speedInput = view.findViewById(R.id.speed_input);
                missingPersonName =
view.findViewById(R.id.missing_person_name_input);
                String missingPersonNameInput =
missingPersonName.getText().toString();
                ownerName = view.findViewById(R.id.search_owner_input);
                String ownerNameInput = ownerName.getText().toString();
                float speedInputValue = 0;
                try {
                    speedInputValue =
Float.parseFloat(speedInput.getText().toString());
                    if(searchNameInput.length() == 0 ||
missingPersonNameInput.length() == 0 || ownerNameInput.length() == 0) {
                        Toast.makeText(getActivity(), "Please fill in all of the
fields", Toast.LENGTH_SHORT).show();
                        if (time == 0) {
                            time = System.currentTimeMillis();
                        }
                    } else {
                        createSearch(searchNameInput, ownerNameInput,
bundle.getDouble("lat", 0), bundle.getDouble("lon", 0), speedInputValue,
missingPersonNameInput, time);
                    }
                } catch (NumberFormatException e) {
                    Toast.makeText(getActivity(), "Please enter a speed",
Toast.LENGTH_SHORT).show();
                }
            }
```

```
        });
    }

    private void createSearch(String searchName, String ownerName, double startLat,
double startLon, float speed, String missingPersonName, long time) {
        Map<String, Object> data = new HashMap<>();
        data.put("searchName", searchName);
        data.put("owner", ownerName);
        data.put("startLat", startLat);
        data.put("startLon", startLon);
        data.put("speed", speed);
        data.put("missingPersonName", missingPersonName);
        data.put("startTime", time);

        onlineDatabase.collection("searches")
                .add(data)
                .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                    @Override
                    public void onSuccess(DocumentReference documentReference) {
                        Toast.makeText(getActivity(), "Search created and saved to
online database", Toast.LENGTH_SHORT).show();
                        MapFragment mapFragment = new MapFragment();
                        Bundle newBundle = new Bundle();
                        newBundle.putBoolean("polygonToPlot", true);
                        newBundle.putFloat("speedInputValue", speed);
                        newBundle.putLong("startTime", time);
                        newBundle.putDouble("lat", lat);
                        newBundle.putDouble("lon", lon);
                        mapFragment.setArguments(newBundle);

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(getActivity(), "Search created but failed to
save to online database", Toast.LENGTH_SHORT).show();
                    }
                });
    }

}
```

CardAdapter.java

```
package com.bengavin.missingpersons;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import java.util.ArrayList;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.recyclerview.widget.RecyclerView;

public class CardAdapter extends RecyclerView.Adapter<CardHolder> {

   Context context;
   ArrayList<CardModel> cardModels;

   public CardAdapter(Context context, ArrayList<CardModel> cardModels) {
       this.context = context;
       this.cardModels = cardModels;
   }

   @NonNull
   @Override
   public CardHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
       View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.search_card, null);
//Inflate search_card.xml when the view holder gets created

       return new CardHolder(view);
   }

   @Override
   public void onBindViewHolder(@NonNull CardHolder holder, int position) {
       holder.mSearchName.setText(cardModels.get(position).getSearchName());
       holder.mSearchOwner.setText(cardModels.get(position).getSearchOwner());

       holder.setCardClickListener(new CardClickListener() {
           //Use interface to make a click listener
           @Override
           public void onCardClickListener(View view, int position) {
               String name = cardModels.get(position).getSearchName();
               String owner =
cardModels.get(position).getSearchOwner().substring(10);
               String uid = cardModels.get(position).getUid();
               //Get attributes from the clicked search card

               Fragment fragment = new SearchInsightFragment();
               Bundle bundle = new Bundle();
               bundle.putString("name", name);
               bundle.putString("owner", owner);
               bundle.putString("uid", uid);
               fragment.setArguments(bundle);
               //This makes a new fragment and puts the attributes retrieved above,
into a bundle which can be used by the secondary fragment
```

```
                FragmentManager fragmentManager =
((AppCompatActivity)context).getSupportFragmentManager();
//((AppCompatActivity)context) used for context due this extending an Adapter
                fragmentManager.beginTransaction().replace(R.id.fragment_container,
fragment).addToBackStack(null).commit(); //Change fragments
            }
        });
    }

    @Override
    public int getItemCount() {
        return cardModels.size();
    }
}
```

CardHolder.java

```
package com.bengavin.missingpersons;

import android.view.View;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class CardHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

    TextView mSearchName, mSearchOwner;
    CardClickListener cardClickListener;

    CardHolder(@NonNull View itemView) {
        super(itemView);

        this.mSearchName = itemView.findViewById(R.id.searchName);
        this.mSearchOwner = itemView.findViewById(R.id.searchOwner);

        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        this.cardClickListener.onCardClickListener(v, getLayoutPosition());
    }

    public void setCardClickListener(CardClickListener cardClickListener) {
        this.cardClickListener = cardClickListener;
    }

}
```

SearchInsightFragment.java

```
package com.bengavin.missingpersons;

import android.os.Bundle;
```

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class SearchInsightFragment extends Fragment {

   private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
   private Button backButton, openMapButton;
   private TextView mSearchName, mSearchOwner, mMissingPersonName, mSpeed,
mStartLatLon, mStartTime;
   private float speed;
   private double startLat, startLon;
   private long time;

   @Nullable
   @Override
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       //When the fragment gets created this will inflate the fragment view
       View fragmentView = inflater.inflate(R.layout.fragment_search_insight,
container, false);
       return fragmentView;
   }

   @Override
   public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
       super.onViewCreated(view, savedInstanceState);

       backButton = view.findViewById(R.id.insightBackButton);
       openMapButton = view.findViewById(R.id.insightOpenMapButton);

       mSearchName = view.findViewById(R.id.insightSearchName);
       mSearchOwner = view.findViewById(R.id.insightSearchOwner);
       mMissingPersonName = view.findViewById(R.id.insightMissingPersonName);
       mSpeed = view.findViewById(R.id.insightSpeed);
       mStartLatLon = view.findViewById(R.id.insightLatLon);
       mStartTime = view.findViewById(R.id.insightStartTime);

       backButton.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               getFragmentManager().popBackStackImmediate();
           }
```

```
        });

        openMapButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MapFragment mapFragment = new MapFragment();
                Bundle newBundle = new Bundle();
                newBundle.putBoolean("polygonToPlot", true);
                newBundle.putFloat("speedInputValue", speed);
                newBundle.putLong("startTime", time);
                newBundle.putDouble("lat", startLat);
                newBundle.putDouble("lon", startLon);
                mapFragment.setArguments(newBundle);

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
            }
        });

        Bundle bundle = this.getArguments(); //Get the arguments passed by
CardAdapter from the bundle
        String uid = bundle.getString("uid", "NO_KEY");
        if (uid.equals("NO_KEY")) {
            Toast.makeText(getActivity(), "An error occurred whilst fetching the
search\nPlease try again later.", Toast.LENGTH_LONG).show();
        } else {
            fetchData(uid, new FirestoreDataCallback() {
                @Override
                public void onData(String searchName, String owner, String
missingPersonName, String speed, String startLat, String startLon, String
startTime) {
                    setSpeed(Float.valueOf(speed));
                    setTime(Long.valueOf(startTime));
                    setStartLat(Double.valueOf(startLat));
                    setStartLon(Double.valueOf(startLon));
                    mSearchName.setText(searchName);
                    mSearchOwner.setText(owner);
                    mMissingPersonName.setText(missingPersonName);
                    mSpeed.setText(speed);
                    mStartLatLon.setText(getString(R.string.lat_lon_template,
startLat, startLon));
                    mStartTime.setText(startTime);
                }
            });
        }
    }

    private void fetchData(String uid, FirestoreDataCallback firestoreDataCallback)
{
        onlineDatabase.collection("searches")
                .document(uid)
                .get()
                .addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                    @Override
```

```
                    public void onSuccess(DocumentSnapshot documentSnapshot) {
                        String searchName =
documentSnapshot.getString("searchName");
                        String owner = documentSnapshot.getString("owner");
                        String missingPersonName =
documentSnapshot.getString("missingPersonName");
                        String speed =
String.valueOf(documentSnapshot.get("speed"));
                        String startLat =
String.valueOf(documentSnapshot.get("startLat"));
                        String startLon =
String.valueOf(documentSnapshot.get("startLon"));
                        String startTime =
String.valueOf(documentSnapshot.get("startTime"));
                        firestoreDataCallback.onData(searchName, owner,
missingPersonName, speed, startLat, startLon, startTime);
                    }
                });
    }

    public double getSpeed() {
        return speed;
    }

    public void setSpeed(float speed) {
        this.speed = speed;
    }

    public double getStartLat() {
        return startLat;
    }

    public void setStartLat(double startLat) {
        this.startLat = startLat;
    }

    public double getStartLon() {
        return startLon;
    }

    public void setStartLon(double startLon) {
        this.startLon = startLon;
    }

    public long getTime() {
        return time;
    }

    public void setTime(long time) {
        this.time = time;
    }
}
```

Iteration 3 - Isochrone with Topography

MapFragment.java

```
package com.bengavin.missingpersons;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.Location;
import android.os.Bundle;
import android.os.Looper;
import android.util.Log;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.Toast;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.mapbox.android.core.location.LocationEngine;
import com.mapbox.android.core.location.LocationEngineCallback;
import com.mapbox.android.core.location.LocationEngineProvider;
import com.mapbox.android.core.location.LocationEngineRequest;
import com.mapbox.android.core.location.LocationEngineResult;
import com.mapbox.geojson.Point;
import com.mapbox.geojson.Polygon;
import com.mapbox.mapboxsdk.camera.CameraUpdateFactory;
import com.mapbox.mapboxsdk.geometry.LatLng;
import com.mapbox.mapboxsdk.geometry.LatLngBounds;
import com.mapbox.mapboxsdk.location.LocationComponent;
import com.mapbox.mapboxsdk.location.LocationComponentActivationOptions;
import com.mapbox.mapboxsdk.location.modes.CameraMode;
import com.mapbox.mapboxsdk.location.modes.RenderMode;
import com.mapbox.mapboxsdk.maps.MapView;
import com.mapbox.mapboxsdk.maps.MapboxMap;
import com.mapbox.mapboxsdk.maps.OnMapReadyCallback;
import com.mapbox.mapboxsdk.maps.Style;
import com.mapbox.mapboxsdk.plugins.markerview.MarkerView;
import com.mapbox.mapboxsdk.plugins.markerview.MarkerViewManager;
import com.mapbox.mapboxsdk.style.layers.FillLayer;
import com.mapbox.mapboxsdk.style.sources.GeoJsonSource;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.lang.ref.WeakReference;
import java.util.ArrayList;
```

```
import java.util.List;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

import static com.mapbox.mapboxsdk.style.layers.PropertyFactory.fillColor;
import static com.mapbox.mapboxsdk.style.layers.PropertyFactory.fillOpacity;

public class MapFragment extends Fragment implements OnMapReadyCallback,
MapboxMap.OnMapClickListener {

   private static final long DEFAULT_INTERVAL = 5000L; //Time interval for location
to be checked
   private static final long DEFAULT_MAX_TIME = DEFAULT_INTERVAL * 5; //Max
interval for location
   private MainActivity activity;
   private MapView mapView;
   private MapboxMap mapboxMap;
   private Context context;
   private LocationEngine locationEngine;
   private MarkerViewManager markerViewManager;
   private MarkerView markerView;
   private double startLat, startLon;
   private LocationChangeListeningActivityLocationCallback callback = new
LocationChangeListeningActivityLocationCallback(this);
   private Bundle startBundle;
   private boolean polygonToPlot = false;

   //Most overridden methods come from the android fragment lifecycle which can be
found here: https://developer.android.com/guide/components/fragments

   @Override
   public void onAttach(Context context) {
       super.onAttach(context);
       Context activity = context;
   }

   @Nullable
   @Override
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       //When the fragment gets created this will inflate the fragment view
       View fragmentView = inflater.inflate(R.layout.fragment_map, container,
false);
       return fragmentView;
   }

   @Override
   public void onViewCreated(View view, Bundle savedInstanceState) {
```

```
        activity = (MainActivity)context;
        super.onViewCreated(view, savedInstanceState);
        mapView = (MapView) view.findViewById(R.id.mapView);
        mapView.onCreate(savedInstanceState);
        mapView.getMapAsync(this);

        startBundle = this.getArguments();
        if(startBundle != null) {
            polygonToPlot = startBundle.getBoolean("polygonToPlot", false);
        }

        final EditText mapSearchBox = (EditText)
view.findViewById(R.id.map_search_bar);
        mapSearchBox.setOnKeyListener(new View.OnKeyListener() { //Listen for key
presses from the search box at the top, if it is the "enter" key, take value from
the EditText and call another method
            @Override
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                if((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode ==
KeyEvent.KEYCODE_ENTER)) {
                    if(mapSearchBox.getText().toString().length() > 0) {
                        addressToCoordinates(mapSearchBox.getText().toString());
//Run addressToCoordinates with value from EditText as string
                    } else {
                        Toast.makeText(getActivity(), "Please enter something into
the search box", Toast.LENGTH_SHORT).show();
                    }
                }
                return false;
            }
        });

        Button startSearch = getActivity().findViewById(R.id.startSearchButton);
        startSearch.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showSearchInput();
            }
        });

        Button clearScreen = getActivity().findViewById(R.id.clearButton);
        clearScreen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                clearScreen();
            }
        });
    }

    @Override
    public void onMapReady(@NonNull MapboxMap mapboxMap) {
        this.mapboxMap = mapboxMap;
        mapboxMap.setStyle(Style.MAPBOX_STREETS, new Style.OnStyleLoaded() {
            @Override
```

```
            public void onStyleLoaded(@NonNull Style style) {
                markerViewManager = new MarkerViewManager(mapView, mapboxMap);
                //enableLocation(style);
                if(polygonToPlot) {
                    float speedInputValue = startBundle.getFloat("speedInputValue",
3);
                    long startTime = startBundle.getLong("startTime",
System.currentTimeMillis());
                    double lat = startBundle.getDouble("lat", 0);
                    double lon = startBundle.getDouble("lon", 0);
                    Isochrone isochrone = new Isochrone(getContext());
                    long time = isochrone.getTimeDifference(startTime);
                    double distance = isochrone.getDistanceTravelled(time,
speedInputValue);
                    double[] pair = {lat, lon};
                    ArrayList<Double> coordinates =
isochrone.getMaxCoordinates(pair, distance);
                    coordinates.add(lat);
                    coordinates.add(lon);
                    isochrone.getElevations(coordinates, new TilequeryCallback() {
                        @Override
                        public void onData(ArrayList<Integer> elevationAtPoints) {
                            Log.e("app", elevationAtPoints.toString());
                            ArrayList<Double> finalCoordinates =
isochrone.applyElevationsToPoints(coordinates, elevationAtPoints, speedInputValue);
                            getActivity().runOnUiThread(() -> {
                                // PLOT ELEVATION SEARCH AREA
                                List<List<Point>> POINTS_ELEVATION = new
ArrayList<>();
                                List<Point> OUTER_POINTS_ELEVATION = new
ArrayList<>();

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(1),
finalCoordinates.get(0)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(3),
finalCoordinates.get(2)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(5),
finalCoordinates.get(4)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(7),
finalCoordinates.get(6)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(9),
finalCoordinates.get(8)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(11),
finalCoordinates.get(10)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(13),
finalCoordinates.get(12)));
```

```
OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(15),
finalCoordinates.get(14)));
                                POINTS_ELEVATION.add(OUTER_POINTS_ELEVATION);
                                style.addSource(new GeoJsonSource("35",
Polygon.fromLngLats(POINTS_ELEVATION)));
                                style.addLayerBelow(new FillLayer("13",
"35").withProperties(
                                        fillColor(Color.parseColor("#ED3742")),
fillOpacity(0.6f)), "style"
                                );
                                LatLngBounds latLngBounds = new
LatLngBounds.Builder()
                                        .include(new LatLng(finalCoordinates.get(0),
finalCoordinates.get(1)))
                                        .include(new LatLng(finalCoordinates.get(2),
finalCoordinates.get(3)))
                                        .include(new LatLng(finalCoordinates.get(4),
finalCoordinates.get(5)))
                                        .include(new LatLng(finalCoordinates.get(6),
finalCoordinates.get(7)))
                                        .include(new LatLng(finalCoordinates.get(8),
finalCoordinates.get(9)))
                                        .include(new
LatLng(finalCoordinates.get(10), finalCoordinates.get(11)))
                                        .include(new
LatLng(finalCoordinates.get(12), finalCoordinates.get(13)))
                                        .include(new
LatLng(finalCoordinates.get(14), finalCoordinates.get(15)))
                                        .build();

mapboxMap.animateCamera(CameraUpdateFactory.newLatLngBounds(latLngBounds, 100),
3000);
                                FloatingActionButton floatingActionButton =
mapView.findViewById(R.id.clearFloatingActionButton);
                                floatingActionButton.show();
                                floatingActionButton.setOnClickListener(new
View.OnClickListener() {
                                    @Override
                                    public void onClick(View v) {
                                        MapFragment mapFragment = new MapFragment();

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
                                    }
                                });
                            });
                        }
                    });
                }
            }
        });
        mapboxMap.addOnMapClickListener(this);
    }
```

```
    @Override
    public boolean onMapClick(@NonNull LatLng point) {
        if(markerView != null) {
            markerViewManager.removeMarker(markerView);
            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            markerView = new MarkerView(point, customView);
            markerViewManager.addMarker(markerView);
            setStartLat(point.getLatitude());
            setStartLon(point.getLongitude());
            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
            view.setVisibility(View.VISIBLE);
        } else {
            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            markerView = new MarkerView(point, customView);
            markerViewManager.addMarker(markerView);
            setStartLat(point.getLatitude());
            setStartLon(point.getLongitude());
            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
            view.setVisibility(View.VISIBLE);
        }
        return true;
    }

    @SuppressWarnings({"MissingPermission"})
    private void enableLocation(@NonNull Style mapStyle) {
        if(ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
            activity = (MainActivity) getContext(); //Essentially replacing "this"
as "this" can't be used in fragments as there is no context passed here
            LocationComponent locationComponent = mapboxMap.getLocationComponent();
            LocationComponentActivationOptions locationComponentActivationOptions =
LocationComponentActivationOptions.builder(activity,
mapStyle).useDefaultLocationEngine(false).build();

locationComponent.activateLocationComponent(locationComponentActivationOptions);
            locationComponent.setLocationComponentEnabled(true);
            locationComponent.setCameraMode(CameraMode.TRACKING);
            //locationComponent.setRenderMode(RenderMode.COMPASS); Removed due to a
known issue with the library here:
https://github.com/mapbox/mapbox-gl-native/issues/14889 - Reported fixed
https://github.com/mapbox/mapbox-gl-native-android/pull/19 with build 8.5.0-beta.1
but error still occurring so it has been commented out
```

```
            initLocationEngine();
        } else {
            requestPermissions(
                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    1
            ); //Request location permission if not already granted
        }
    }

    @SuppressWarnings({"MissingPermission"})
    private void initLocationEngine() {
        locationEngine = LocationEngineProvider.getBestLocationEngine(activity);
//Initialise new location engine
        LocationEngineRequest request = new
LocationEngineRequest.Builder(DEFAULT_INTERVAL)
                .setPriority(LocationEngineRequest.PRIORITY_HIGH_ACCURACY)
                .setMaxWaitTime(DEFAULT_MAX_TIME).build(); //Set time intervals for
updating user location (2s)
        locationEngine.requestLocationUpdates(request, callback,
Looper.getMainLooper());
        locationEngine.getLastLocation(callback); //Pass it to callback (ie
LocationChangeListeningActivityLocationCallback)
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        //Method called when requestPermissions(); gets called
        if(requestCode == 1) {
            if (permissions[0].equals(Manifest.permission.ACCESS_FINE_LOCATION) &&
grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                //Check if location permissions have been granted
                mapboxMap.getStyle(new Style.OnStyleLoaded() {
                    @Override
                    public void onStyleLoaded(@NonNull Style style) {
                        enableLocation(style); //Rerun method, this time with
correct permissions
                    }
                });
            }
        } else {
            Toast.makeText(getActivity(), R.string.location_access_not_permitted,
Toast.LENGTH_LONG).show(); //Let user know they need to allow location permission
        }
    }

    private void addressToCoordinates(@NonNull String address) {
        address = address.replaceAll("/[^a-zA-Z0-9 ]/g", "").replaceAll(" ", "%20");
        String url = "https://api.mapbox.com/geocoding/v5/mapbox.places/" + address
+ ".json?&access_token=" + getString(R.string.mapbox_token);
        //Variables must be declared before starting a thread as they have to be
final or effectively-final
        //Creating a thread - running network operations must happen on another
thread as the main/UI thread can skip frames or lifecycle methods
```

```
       Thread thread = new Thread(() -> {
           Thread.currentThread().setPriority(Thread.MIN_PRIORITY); //Important to
set the thread priority to MIN or at least less than main/UI thread to prevent
skipping frames or lifecycle methods
           OkHttpClient okHttpClient = new OkHttpClient(); //Initialising an
instance of the HTTP client
           Request request = new Request.Builder() //Build request with URL and
optional headers (not needed)
                   .url(url)
                   .build();
           Response response = null; //Cannot run query in a try(query here){} due
to a target API mismatch (so it runs on more devices)
           try {
               response = okHttpClient.newCall(request).execute();
               String responseStr = response.body().string();
               JSONObject jsonObject = new JSONObject(responseStr); //Convert the
string into a JSONObject for manipulation and data reading
               JSONArray allFeatures = jsonObject.getJSONArray("features");
               JSONObject firstObject = allFeatures.getJSONObject(0);
               JSONObject geometryObject = firstObject.getJSONObject("geometry");
               JSONArray coordPair = geometryObject.getJSONArray("coordinates");
//Final coordinate pair taken from API response in order to plot marker on the map
               double lat = ((Number) coordPair.get(1)).doubleValue();
               double lng = ((Number) coordPair.get(0)).doubleValue();
               setStartLat(lat);
               setStartLon(lng);
               getActivity().runOnUiThread(new Runnable() {
                   @Override
                   public void run() {
                       if(markerView != null) {
                           markerViewManager.removeMarker(markerView);
                           View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
                           customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                           markerView = new MarkerView(new LatLng(lat, lng),
customView);
                           markerViewManager.addMarker(markerView);
                           ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
                           view.setVisibility(View.VISIBLE);
                       } else {
                           View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
                           customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                           markerView = new MarkerView(new LatLng(lat, lng),
customView);
                           markerViewManager.addMarker(markerView);
                           ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
                           view.setVisibility(View.VISIBLE);
```

```
                    }
                }
            });
        } catch (JSONException e) {
            Toast.makeText(getActivity(), "Failed to plot point\nPlease try
again later", Toast.LENGTH_LONG).show();
        } catch(IOException e) {
            Toast.makeText(getActivity(), R.string.ioexception_error_message,
Toast.LENGTH_LONG).show();
        } finally {
            if(response != null) {
                response.body().close(); //Close the client after checking it
existed in the first place
            }
        }
    });
    thread.start();
}

private void clearScreen() {
    View view = (View) getActivity().findViewById(R.id.startSearchRectangle);
    if (view.getVisibility()== View.VISIBLE) {
        view.setVisibility(View.GONE);
    }
    markerViewManager.removeMarker(markerView);
    final EditText mapSearchBox = (EditText)
getActivity().findViewById(R.id.map_search_bar);
    mapSearchBox.getText().clear();
}

private void showSearchInput() {
    View view = (View) getActivity().findViewById(R.id.startSearchRectangle);
    if (view.getVisibility() == View.VISIBLE) {
        view.setVisibility(View.GONE);
    }
    Fragment newSearchFragment = new NewSearchFragment();
    Bundle bundle = new Bundle();
    bundle.putDouble("lat", startLat);
    bundle.putDouble("lon", startLon);
    newSearchFragment.setArguments(bundle);
    getFragmentManager().beginTransaction().replace(R.id.fragment_container,
newSearchFragment).commit();
}

private static class LocationChangeListeningActivityLocationCallback implements
LocationEngineCallback<LocationEngineResult> {

    private final WeakReference<MapFragment> fragmentWeakReference;

    LocationChangeListeningActivityLocationCallback(MapFragment fragment) {
        this.fragmentWeakReference = new WeakReference<>(fragment);
    }

    @Override
```

```
        public void onSuccess(LocationEngineResult result) {
            MapFragment fragment = fragmentWeakReference.get();

            if(fragment != null) {
                Location location = result.getLastLocation(); //Fetches last
location

                if(location == null) {
                    return;
                }
                if(fragment.mapboxMap != null && result.getLastLocation() != null) {

fragment.mapboxMap.getLocationComponent().forceLocationUpdate(result.getLastLocatio
n()); //Forces map to update user location to last lat lon from location engine
                }
            }
        }

        @Override
        public void onFailure(@NonNull Exception ignored) {

        }

    }

    private double getStartLat() {
        return this.startLat;
    }

    private void setStartLat(double lat) {
        this.startLat = lat;
    }

    private double getStartLon() {
        return this.startLon;
    }

    private void setStartLon(double lon) {
        this.startLon = lon;
    }

    @SuppressWarnings({"MissingPermission"})
    @Override
    public void onStart() {
        super.onStart();
        mapView.onStart();
    }

    @Override
    public void onSaveInstanceState(@NonNull Bundle outState) {
        super.onSaveInstanceState(outState);
        mapView.onSaveInstanceState(outState);
    }
```

```
    @Override
    public void onDestroyView() {
        super.onDestroyView();
        if(locationEngine != null) {
            locationEngine.removeLocationUpdates(callback);
        }
        if(mapView != null) {
            mapView.onDestroy();
        }
    }

    @Override
    public void onPause() {
        super.onPause();
        mapView.onPause();
    }

    @Override
    public void onResume() {
        super.onResume();
        mapView.onResume();
    }

}
```

Isochrone.java

```
package com.bengavin.missingpersons;

import android.content.Context;

import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.NoSuchElementException;

import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class Isochrone {

    private Context context;
    private volatile ArrayList<Integer> elevationsAtPoints = new ArrayList<>();
    private int HEIGHT_PER_MINUTE = 10; // For every 10 metres travelled in
elevation  ≈ 1 minute of travel

    public Isochrone(Context context) {
        this.context = context;
```

```
    }

    private double toRadians(double degrees) {
        double radians = degrees * ((Math.PI)/180);
        return radians;
    }

    private double toDegrees(double radians) {
        double factor = 180 / Math.PI;
        return radians * factor;
    }

    public ArrayList<Double> getMaxCoordinates(double[] startingPair, double
distance) {
        double lat = toRadians(startingPair[0]);
        double lon = toRadians(startingPair[1]);
        double distanceRatio = distance/6371.01; //Divide the distance given by the
radius of the Earth to get a ratio
        ArrayList<Double> endMaxCoordinates = new ArrayList<Double>(); //Initialise
end array for all coordinates (size 16 as 8 pairs)
        for(int i = 0; i < 360; i+= 45) { // <360 as 360 should not be included as
it is the same as 0
            double bearing = toRadians(i);
            double endLat = Math.asin(Math.sin(lat)*Math.cos(distanceRatio) +
Math.cos(lat)*Math.sin(distanceRatio)*Math.cos(bearing));
            endMaxCoordinates.add(toDegrees(endLat));
            double endLon = lon +
Math.atan2(Math.sin(bearing)*Math.sin(distanceRatio)*Math.cos(lat),
Math.cos(distanceRatio)-Math.sin(lat)*Math.sin(endLat));
            endMaxCoordinates.add(toDegrees(endLon));
        }
        return endMaxCoordinates;
    }

    public long getTimeDifference(long startTime) {
        startTime = startTime/1000L;
        long unixTime = System.currentTimeMillis()/1000L;
        long difference = unixTime - startTime;
        return difference;
    }

    public double getDistanceTravelled(double timeDifference, double speed) {
        timeDifference = timeDifference/60/60;  // Gets time in hours
        double displacement = timeDifference * speed;
        return displacement; // Displace in km/h
    }

    public void getElevations(ArrayList<Double> coordinates, TilequeryCallback
tilequeryCallback) {
        Thread thread = new Thread(() -> {
            int j = 0;
            int k = 1;
            for (int i = 0; i < 9; i++) {
```

```
                String url =
"https://api.mapbox.com/v4/mapbox.mapbox-terrain-v2/tilequery/" +
coordinates.get(k) + "," + coordinates.get(j) +
".json?layers=contour&limit=50&access_token=" +
context.getString(R.string.mapbox_token);
                Thread.currentThread().setPriority(Thread.MIN_PRIORITY);
                OkHttpClient okHttpClient = new OkHttpClient();
                Request request = new Request.Builder()
                        .url(url)
                        .build();
                Response response = null;
                try {
                    response = okHttpClient.newCall(request).execute();
                    String responseStr = response.body().string();
                    JSONObject jsonObject = new JSONObject(responseStr);
                    JSONArray allFeatures = jsonObject.getJSONArray("features");
                    ArrayList<Integer> elevations = new ArrayList<>();
                    for (int m = 0; m < allFeatures.length(); m++) {
                        JSONObject feature = allFeatures.getJSONObject(m);
                        JSONObject properties = feature.getJSONObject("properties");
                        int featureElevation = (Integer) properties.get("ele"); //
API always returns an int
                        elevations.add(featureElevation);
                    }
                    int highestElevation = 0;
                    try {
                        highestElevation = Collections.max(elevations); // Get max
elevation
                    } catch (NoSuchElementException e) {
                        Toast.makeText(context, "The elevation for point " + i + "
failed to load - using max possible distance", Toast.LENGTH_SHORT).show();
                    }
                    elevationsAtPoints.add(highestElevation);
                } catch (JSONException e) {
                    Toast.makeText(context, "Failed to plot point\nPlease try again
later", Toast.LENGTH_LONG).show();
                } catch (IOException e) {
                    Toast.makeText(context,
context.getString(R.string.ioexception_error_message), Toast.LENGTH_LONG).show();
                } finally {
                    if(response != null) {
                        response.body().close();
                    }
                }
                j = j + 2;
                k = k + 2;
                if(i == 8) {
                    tilequeryCallback.onData(elevationsAtPoints);
                }
            }
        });
        thread.start();
    }
```

```
   private ArrayList<Double> retractDistance(int bearing, double lat, double lon,
double distance) {
       lat = toRadians(lat);
       lon = toRadians(lon);
       double oppositeBearing = toRadians(360 - bearing); // Invert
bearing/direction
       double distanceRatio = distance/6371.01;
       ArrayList<Double> endCoordinates = new ArrayList<>();
       double endLat = Math.asin(Math.sin(lat)*Math.cos(distanceRatio) +
Math.cos(lat)*Math.sin(distanceRatio)*Math.cos(oppositeBearing));
       endCoordinates.add(toDegrees(endLat));
       double endLon = lon +
Math.atan2(Math.sin(oppositeBearing)*Math.sin(distanceRatio)*Math.cos(lat),
Math.cos(distanceRatio)-Math.sin(lat)*Math.sin(endLat));
       endCoordinates.add(toDegrees(endLon));
       return endCoordinates;
   }

   public ArrayList<Double> applyElevationsToPoints(ArrayList<Double>
pointCoordinates, ArrayList<Integer> elevations, double speed) {
       int j = 0;
       int k = 1;
       int elevationAtStartPoint = elevations.get(elevations.size()-1);
       ArrayList<Double> endCoordinates = new ArrayList<>();
       for(int i = 0; i < pointCoordinates.size()/2; i++) {
           int bearing = 0;
           double lat = pointCoordinates.get(j);
           double lon = pointCoordinates.get(k);
           int elevationAtPoint = elevations.get(i);
           int heightDifference = elevationAtStartPoint - elevationAtPoint;
           int minutesToAdd = Math.abs(Math.round(heightDifference /
HEIGHT_PER_MINUTE));
           double distanceTravelled = getDistanceTravelled(minutesToAdd * 60,
speed);
           ArrayList<Double> retractionCoordinates = retractDistance(bearing, lat,
lon, distanceTravelled);
           double endLat = retractionCoordinates.get(0);
           endCoordinates.add(endLat);
           double endLon = retractionCoordinates.get(1);
           endCoordinates.add(endLon);
           j = j + 2;
           k = k + 2;
           bearing = bearing + 45;
       }
       return endCoordinates;
   }

}
```

TilequeryCallback.java

```
package com.bengavin.missingpersons;

import java.util.ArrayList;
```

```
public interface TilequeryCallback {

    void onData(ArrayList<Integer> elevationAtPoints);

}
```

Iteration 4 - Deleting Searches

NewSearchFragment.java

```
package com.bengavin.missingpersons;

import android.app.AlertDialog;
import android.os.Bundle;
import android.provider.Settings;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.mapbox.mapboxsdk.geometry.LatLng;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class NewSearchFragment extends Fragment {

    private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
    private Button cancelButton, submitButton, setDateTimeButton;
    private EditText searchName, speedInput, missingPersonName, ownerName;
    private TextView timeOutput;
    private Bundle bundle;
    private long time = 0;
    private double lat, lon;

    @Nullable
    @Override
```

```
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       View fragmentView = inflater.inflate(R.layout.fragment_new_search,
container, false);
       return fragmentView;
   }

   @Override
   public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
       super.onViewCreated(view, savedInstanceState);

       setDateTimeButton = view.findViewById(R.id.set_date_time);
       cancelButton = view.findViewById(R.id.cancel_button);
       submitButton = view.findViewById(R.id.submit_button);
       timeOutput = view.findViewById(R.id.time_picker_output);

       bundle = this.getArguments();
       lat = bundle.getDouble("lat", 0);
       lon = bundle.getDouble("lon", 0);

       setDateTimeButton.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               final View dialogView = View.inflate(getActivity(),
R.layout.date_time_picker, null);
               final AlertDialog alertDialog = new
AlertDialog.Builder(getActivity()).create();

               dialogView.findViewById(R.id.set).setOnClickListener(new
View.OnClickListener() {
                   @Override
                   public void onClick(View inner) {
                       DatePicker datePicker =
alertDialog.findViewById(R.id.date_picker);
                       TimePicker timePicker =
alertDialog.findViewById(R.id.time_picker);

                       timePicker.setIs24HourView(true);

                       Calendar calendar = new
GregorianCalendar(datePicker.getYear(), datePicker.getMonth(),
datePicker.getDayOfMonth(), timePicker.getCurrentHour(),
timePicker.getCurrentMinute());

                       time = calendar.getTimeInMillis();

timeOutput.setText(getString(R.string.date_time_output_template,
String.valueOf(timePicker.getCurrentHour()),
String.valueOf(timePicker.getCurrentMinute()),
String.valueOf(datePicker.getDayOfMonth()), String.valueOf(datePicker.getMonth()),
String.valueOf(datePicker.getYear())));
                       alertDialog.dismiss();
                   }
               });
```

```
                alertDialog.setView(dialogView);
                alertDialog.show();
            }
        });

        cancelButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

getFragmentManager().beginTransaction().replace(R.id.fragment_container, new
MapFragment()).commit();
            }
        });

        submitButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                searchName = view.findViewById(R.id.search_name_input);
                String searchNameInput = searchName.getText().toString();
                speedInput = view.findViewById(R.id.speed_input);
                missingPersonName =
view.findViewById(R.id.missing_person_name_input);
                String missingPersonNameInput =
missingPersonName.getText().toString();
                ownerName = view.findViewById(R.id.search_owner_input);
                String ownerNameInput = ownerName.getText().toString();
                float speedInputValue = 0;
                try {
                    speedInputValue =
Float.parseFloat(speedInput.getText().toString());
                    if(searchNameInput.length() == 0 ||
missingPersonNameInput.length() == 0 || ownerNameInput.length() == 0) {
                        Toast.makeText(getActivity(), "Please fill in all of the
fields", Toast.LENGTH_SHORT).show();
                        if (time == 0) {
                            time = System.currentTimeMillis();
                        }
                    } else {
                        createSearch(searchNameInput, ownerNameInput,
bundle.getDouble("lat", 0), bundle.getDouble("lon", 0), speedInputValue,
missingPersonNameInput, time);
                    }
                } catch (NumberFormatException e) {
                    Toast.makeText(getActivity(), "Please enter a speed",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private void createSearch(String searchName, String ownerName, double startLat,
double startLon, float speed, String missingPersonName, long time) {
```

```
        String androidUID =
Settings.Secure.getString(getContext().getContentResolver(),
Settings.Secure.ANDROID_ID);

        Map<String, Object> data = new HashMap<>();
        data.put("searchName", searchName);
        data.put("owner", ownerName);
        data.put("ownerID", androidUID);
        data.put("startLat", startLat);
        data.put("startLon", startLon);
        data.put("speed", speed);
        data.put("missingPersonName", missingPersonName);
        data.put("startTime", time);

        onlineDatabase.collection("searches")
                .add(data)
                .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                    @Override
                    public void onSuccess(DocumentReference documentReference) {
                        Toast.makeText(getActivity(), "Search created and saved to
online database", Toast.LENGTH_SHORT).show();
                        MapFragment mapFragment = new MapFragment();
                        Bundle newBundle = new Bundle();
                        newBundle.putBoolean("polygonToPlot", true);
                        newBundle.putFloat("speedInputValue", speed);
                        newBundle.putLong("startTime", time);
                        newBundle.putDouble("lat", lat);
                        newBundle.putDouble("lon", lon);
                        mapFragment.setArguments(newBundle);

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(getActivity(), "Search created but failed to
save to online database", Toast.LENGTH_SHORT).show();
                    }
                });
    }

}
```

SearchInsightFragment.java

```
package com.bengavin.missingpersons;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.provider.Settings;
import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class SearchInsightFragment extends Fragment {

   private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
   private Button backButton, openMapButton;
   private TextView mSearchName, mSearchOwner, mMissingPersonName, mSpeed,
mStartLatLon, mStartTime;
   private float speed;
   private double startLat, startLon;
   private long time;

   @Nullable
   @Override
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       //When the fragment gets created this will inflate the fragment view
       View fragmentView = inflater.inflate(R.layout.fragment_search_insight,
container, false);
       return fragmentView;
   }

   @SuppressWarnings({"HardwareIds"})
   @Override
   public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
       super.onViewCreated(view, savedInstanceState);

       backButton = view.findViewById(R.id.insightBackButton);
       openMapButton = view.findViewById(R.id.insightOpenMapButton);

       mSearchName = view.findViewById(R.id.insightSearchName);
       mSearchOwner = view.findViewById(R.id.insightSearchOwner);
       mMissingPersonName = view.findViewById(R.id.insightMissingPersonName);
       mSpeed = view.findViewById(R.id.insightSpeed);
       mStartLatLon = view.findViewById(R.id.insightLatLon);
       mStartTime = view.findViewById(R.id.insightStartTime);
```

```
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getFragmentManager().popBackStackImmediate();
            }
        });

        openMapButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MapFragment mapFragment = new MapFragment();
                Bundle newBundle = new Bundle();
                newBundle.putBoolean("polygonToPlot", true);
                newBundle.putFloat("speedInputValue", speed);
                newBundle.putLong("startTime", time);
                newBundle.putDouble("lat", startLat);
                newBundle.putDouble("lon", startLon);
                mapFragment.setArguments(newBundle);

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
            }
        });

        Bundle bundle = this.getArguments(); //Get the arguments passed by
CardAdapter from the bundle
        String uid = bundle.getString("uid", "NO_KEY");
        if (uid.equals("NO_KEY")) {
            Toast.makeText(getActivity(), "An error occurred whilst fetching the
search\nPlease try again later.", Toast.LENGTH_LONG).show();
        } else {
            fetchData(uid, new FirestoreDataCallback() {
                @Override
                public void onData(String searchName, String owner, String ownerID,
String missingPersonName, String speed, String startLat, String startLon, String
startTime, String documentUID) {
                    setSpeed(Float.valueOf(speed));
                    setTime(Long.valueOf(startTime));
                    setStartLat(Double.valueOf(startLat));
                    setStartLon(Double.valueOf(startLon));
                    Date date = new Date(Long.valueOf(startTime));
                    // Put time back into a readable format for the user
                    SimpleDateFormat simpleDateFormat = new SimpleDateFormat("HH:mm
dd-MM-yyyy z", Locale.getDefault());
                    String dateString = simpleDateFormat.format(date);
                    mSearchName.setText(searchName);
                    mSearchOwner.setText(owner);
                    mMissingPersonName.setText(missingPersonName);

mSpeed.setText(String.valueOf(BigDecimal.valueOf(Float.valueOf(speed)).setScale(3,
RoundingMode.HALF_UP).floatValue()));
                    mStartLatLon.setText(getString(R.string.lat_lon_template,
startLat, startLon));
```

```
                    mStartTime.setText(dateString);

                    String androidUID =
Settings.Secure.getString(getContext().getContentResolver(),
Settings.Secure.ANDROID_ID);
                    if(androidUID.equals(ownerID)) {
                        Button deleteButton =
view.findViewById(R.id.insightDeleteButton);
                        deleteButton.setVisibility(View.VISIBLE);
                        deleteButton.setOnClickListener(new View.OnClickListener() {
                            @Override
                            public void onClick(View v) {
                                new AlertDialog.Builder(getContext())
                                        .setTitle("Confirm action")
                                        .setMessage("Are you sure you want to delete
this search? This action cannot be undone.")
                                        .setIcon(R.drawable.ic_error_black_24dp)
                                        .setPositiveButton(R.string.yes, new
DialogInterface.OnClickListener() {
                                            @Override
                                            public void onClick(DialogInterface
dialog, int which) {

onlineDatabase.collection("searches")
                                                    .document(documentUID)
                                                    .delete()
                                                    .addOnSuccessListener(new
OnSuccessListener<Void>() {
                                                        @Override
                                                        public void
onSuccess(Void aVoid) {

Toast.makeText(getActivity(), "Successfully deleted search",
Toast.LENGTH_SHORT).show();

getFragmentManager().popBackStackImmediate();
                                                        }
                                                    })
                                                    .addOnFailureListener(new
OnFailureListener() {
                                                        @Override
                                                        public void
onFailure(@NonNull Exception e) {

Toast.makeText(getActivity(), "Failed to delete search\nPlease try again later",
Toast.LENGTH_SHORT).show();
                                                        }
                                                    });
                                            }
                                        })
                                        .setNegativeButton(R.string.no, null)
                                        .show();
                            }
                        });
```

```
                    }
                }
            });
        }
    }

    private void fetchData(String uid, FirestoreDataCallback firestoreDataCallback)
{
        onlineDatabase.collection("searches")
                .document(uid)
                .get()
                .addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                    @Override
                    public void onSuccess(DocumentSnapshot documentSnapshot) {
                        String searchName =
documentSnapshot.getString("searchName");
                        String owner = documentSnapshot.getString("owner");
                        String ownerID = documentSnapshot.getString("ownerID");
                        String missingPersonName =
documentSnapshot.getString("missingPersonName");
                        String speed =
String.valueOf(documentSnapshot.get("speed"));
                        String startLat =
String.valueOf(documentSnapshot.get("startLat"));
                        String startLon =
String.valueOf(documentSnapshot.get("startLon"));
                        String startTime =
String.valueOf(documentSnapshot.get("startTime"));
                        String documentUID = documentSnapshot.getId();
                        firestoreDataCallback.onData(searchName, owner, ownerID,
missingPersonName, speed, startLat, startLon, startTime, documentUID);
                    }
                });
    }

    public double getSpeed() {
        return speed;
    }

    public void setSpeed(float speed) {
        this.speed = speed;
    }

    public double getStartLat() {
        return startLat;
    }

    public void setStartLat(double startLat) {
        this.startLat = startLat;
    }

    public double getStartLon() {
        return startLon;
    }
```

```
    public void setStartLon(double startLon) {
        this.startLon = startLon;
    }

    public long getTime() {
        return time;
    }

    public void setTime(long time) {
        this.time = time;
    }
}
```

Final Code

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bengavin.missingpersons">

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

    <application
        android:name="com.bengavin.missingpersons.ApplicationClass"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name="com.bengavin.missingpersons.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <action android:name="android.intent.action.VIEW"/>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

strings.xml

```
<resources>
    <string name="app_name">Missing Persons Coordinator</string>
    <string name="nav_map_title">Map</string>
    <string name="nav_searches_title">Current Searches</string>
    <string name="map_search_text">Enter an address here</string>

    <string name="clear_button_text">Clear</string>
    <string name="start_search_button_text">Start Search</string>

    <string name="share_preference">Share with others?</string>
    <string-array name="boolean_input">
```

```
        <item>No</item>
        <item>Yes</item>
    </string-array>
    <string name="speed">Speed (km/h)</string>
    <string name="search_name">Search name</string>
    <string name="missing_person_name">Missing Person Name</string>
    <string name="search_owner_name">Owner Name</string>
    <string name="search_start_date_time">Search Start Time</string>
    <string name="set_search_start_date_time">Set Search Start</string>
    <string name="date_time_output_template">%1$s:%2$s - %3$s/%4$s/%5$s</string>
    <string name="lat_lon">Latitude/Longitude</string>
    <string name="lat_lon_template">%1$s / %2$s</string>

    <string name="yes">Yes</string>
    <string name="no">No</string>
    <string name="submit">Submit</string>
    <string name="cancel">Cancel</string>
    <string name="create">Create</string>
    <string name="open_map">Open Map</string>
    <string name="back">Back</string>
    <string name="delete">Delete</string>

    <string name="search_name_placeholder">Search</string>
    <string name="search_owner_placeholder">Unknown</string>

    <string name="location_access_not_permitted">Please enable location access in
order to plot your location</string>
    <string name="ioexception_error_message">Failed to connect to the internet,
please try again.</string>

    <string name="mapbox_mapsdk_token">[token]</string>
    <string name="mapbox_token">[token]</string>
</resources>
```

styles.xml

```
<resources>

    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#ED3742</color>
    <color name="colorPrimaryDark">#A5262E</color>
    <color name="colorAccent">#A6CE38</color>
</resources>
```

build.gradle (inc dependencies)

```
apply plugin: 'com.android.application'

android {
   compileSdkVersion 28
   defaultConfig {
       applicationId "com.bengavin.missingpersons"
       minSdkVersion 16
       targetSdkVersion 28
       multiDexEnabled true
       versionCode 1
       versionName "1.0"
       testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
   }
   buildTypes {
       release {
           minifyEnabled true
           proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
       }
   }
   compileOptions {
       targetCompatibility = 1.8
       sourceCompatibility = 1.8
   }
}

allprojects {
   gradle.projectsEvaluated {
       tasks.withType(JavaCompile) {
           options.compilerArgs << "-Xlint:deprecation"
       }
   }
}

dependencies {
   implementation fileTree(dir: 'libs', include: ['*.jar'])
   implementation 'androidx.appcompat:appcompat:1.1.0'
   implementation 'com.android.support:design:28.0.0'
   implementation 'com.android.support:recyclerview-v7:28.0.0'
   implementation 'com.android.support:cardview-v7:28.0.0'
   implementation 'com.google.android.material:material:1.0.0'
   implementation 'com.mapbox.mapboxsdk:mapbox-android-sdk:8.5.0'
   implementation 'com.mapbox.mapboxsdk:mapbox-android-plugin-annotation-v8:0.7.0'
   implementation 'com.mapbox.mapboxsdk:mapbox-android-plugin-markerview-v8:0.3.0'
   implementation 'com.google.firebase:firebase-firestore:21.2.1'
   implementation 'androidx.multidex:multidex:2.0.1'
   testImplementation 'junit:junit:4.12'
   androidTestImplementation 'androidx.test:runner:1.2.0'
   androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}

apply plugin: 'com.google.gms.google-services'
```

ApplicationClass.java

```
package com.bengavin.missingpersons;

import com.mapbox.mapboxsdk.Mapbox;

import androidx.multidex.MultiDexApplication;

public class ApplicationClass extends MultiDexApplication {

   //Referenced in manifest
   //Initialises map in the onCreate method
   @Override
   public void onCreate() {
       super.onCreate();
       String token = getString(R.string.mapbox_mapsdk_token);
       Mapbox.getInstance(getApplicationContext(), token);
   }

}
```

MainActivity.java

```
package com.bengavin.missingpersons;

import android.os.Bundle;
import android.view.MenuItem;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);

       setContentView(R.layout.activity_main); //Show the main activity which will
contain all of the fragments

       BottomNavigationView bottomNav = findViewById(R.id.bottom_navigation);
       bottomNav.setOnNavigationItemSelectedListener(bottomNavListener); //Set a
listener to listen for when the tabs are changed at the bottom nav bar


getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new
MapFragment()).commit(); //Start the app by showing the map fragment
   }

   private BottomNavigationView.OnNavigationItemSelectedListener bottomNavListener
= new BottomNavigationView.OnNavigationItemSelectedListener() {
       @Override
       public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
```

```
            Fragment currentFragment = null;
            switch (menuItem.getItemId()) {
                case R.id.nav_map:
                    //Map tab
                    currentFragment = new MapFragment();
                    break;
                case R.id.nav_searches:
                    //Searches tab
                    currentFragment = new SearchesFragment();
                    break;
            }

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
currentFragment).commit();
            return true;
        }
    };

    protected void onDestroy() {
        super.onDestroy();
    }

}
```

MapFragment.java

```
package com.bengavin.missingpersons;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.Location;
import android.os.Bundle;
import android.os.Looper;
import android.util.Log;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.Toast;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.mapbox.android.core.location.LocationEngine;
import com.mapbox.android.core.location.LocationEngineCallback;
import com.mapbox.android.core.location.LocationEngineProvider;
import com.mapbox.android.core.location.LocationEngineRequest;
import com.mapbox.android.core.location.LocationEngineResult;
import com.mapbox.geojson.Point;
import com.mapbox.geojson.Polygon;
import com.mapbox.mapboxsdk.camera.CameraUpdateFactory;
import com.mapbox.mapboxsdk.geometry.LatLng;
```

```
import com.mapbox.mapboxsdk.geometry.LatLngBounds;
import com.mapbox.mapboxsdk.location.LocationComponent;
import com.mapbox.mapboxsdk.location.LocationComponentActivationOptions;
import com.mapbox.mapboxsdk.location.modes.CameraMode;
import com.mapbox.mapboxsdk.location.modes.RenderMode;
import com.mapbox.mapboxsdk.maps.MapView;
import com.mapbox.mapboxsdk.maps.MapboxMap;
import com.mapbox.mapboxsdk.maps.OnMapReadyCallback;
import com.mapbox.mapboxsdk.maps.Style;
import com.mapbox.mapboxsdk.plugins.markerview.MarkerView;
import com.mapbox.mapboxsdk.plugins.markerview.MarkerViewManager;
import com.mapbox.mapboxsdk.style.layers.FillLayer;
import com.mapbox.mapboxsdk.style.sources.GeoJsonSource;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.lang.ref.WeakReference;
import java.util.ArrayList;
import java.util.List;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

import static com.mapbox.mapboxsdk.style.layers.PropertyFactory.fillColor;
import static com.mapbox.mapboxsdk.style.layers.PropertyFactory.fillOpacity;

public class MapFragment extends Fragment implements OnMapReadyCallback,
MapboxMap.OnMapClickListener {

   private static final long DEFAULT_INTERVAL = 5000L; //Time interval for location
to be checked
   private static final long DEFAULT_MAX_TIME = DEFAULT_INTERVAL * 5; //Max
interval for location
   private MainActivity activity;
   private MapView mapView;
   private MapboxMap mapboxMap;
   private Context context;
   private LocationEngine locationEngine;
   private MarkerViewManager markerViewManager;
   private MarkerView markerView;
   private double startLat, startLon;
   private LocationChangeListeningActivityLocationCallback callback = new
LocationChangeListeningActivityLocationCallback(this);
   private Bundle startBundle;
   private boolean polygonToPlot = false;
```

```
   //Most overridden methods come from the android fragment lifecycle which can be
found here: https://developer.android.com/guide/components/fragments

   @Override
   public void onAttach(Context context) {
       super.onAttach(context);
       Context activity = context;
   }

   @Nullable
   @Override
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       //When the fragment gets created this will inflate the fragment view
       View fragmentView = inflater.inflate(R.layout.fragment_map, container,
false);
       return fragmentView;
   }

   @Override
   public void onViewCreated(View view, Bundle savedInstanceState) {
       activity = (MainActivity)context;
       super.onViewCreated(view, savedInstanceState);
       mapView = (MapView) view.findViewById(R.id.mapView);
       mapView.onCreate(savedInstanceState);
       mapView.getMapAsync(this);

       startBundle = this.getArguments();
       if(startBundle != null) {
           polygonToPlot = startBundle.getBoolean("polygonToPlot", false); // Fetch
key value in bundle to see if polygon needs to be plotted
       }

       final EditText mapSearchBox = (EditText)
view.findViewById(R.id.map_search_bar);
       mapSearchBox.setOnKeyListener(new View.OnKeyListener() { //Listen for key
presses from the search box at the top, if it is the "enter" key, take value from
the EditText and call another method
           @Override
           public boolean onKey(View v, int keyCode, KeyEvent event) {
               if((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode ==
KeyEvent.KEYCODE_ENTER)) {
                   if(mapSearchBox.getText().toString().length() > 0) {
                       addressToCoordinates(mapSearchBox.getText().toString());
//Run addressToCoordinates with value from EditText as string
                   } else {
                       Toast.makeText(getActivity(), "Please enter something into
the search box", Toast.LENGTH_SHORT).show();
                   }
               }
               return false;
           }
       });
```

```
        Button startSearch = getActivity().findViewById(R.id.startSearchButton);
        startSearch.setOnClickListener(new View.OnClickListener() { // Sets a click
listener for the start search button
            @Override
            public void onClick(View v) {
                showSearchInput();
            }
        });

        Button clearScreen = getActivity().findViewById(R.id.clearButton);
        clearScreen.setOnClickListener(new View.OnClickListener() { // Sets a click
listener for the cancel button which will remove all markers on the map
            @Override
            public void onClick(View v) {
                clearScreen();
            }
        });
    }

    @Override
    public void onMapReady(@NonNull MapboxMap mapboxMap) {
        this.mapboxMap = mapboxMap;
        mapboxMap.setStyle(Style.MAPBOX_STREETS, new Style.OnStyleLoaded() {
            @Override
            public void onStyleLoaded(@NonNull Style style) {
                markerViewManager = new MarkerViewManager(mapView, mapboxMap); //
Allows markers to be plot easily
                enableLocation(style);
                if(polygonToPlot) {
                    float speedInputValue = startBundle.getFloat("speedInputValue",
3);
                    long startTime = startBundle.getLong("startTime",
System.currentTimeMillis());
                    double lat = startBundle.getDouble("lat", 0);
                    double lon = startBundle.getDouble("lon", 0);
                    Isochrone isochrone = new Isochrone(getContext());
                    long time = isochrone.getTimeDifference(startTime);
                    double distance = isochrone.getDistanceTravelled(time,
speedInputValue);
                    double[] pair = {lat, lon};
                    ArrayList<Double> coordinates =
isochrone.getMaxCoordinates(pair, distance); // Work out maximum coordinates using
speed = distance / time
                    coordinates.add(lat);
                    coordinates.add(lon);
                    isochrone.getElevations(coordinates, new TilequeryCallback() {
// Fetches the elevations for the isochrone, uses a callback to ensure data arrives
                        @Override
                        public void onData(ArrayList<Integer> elevationAtPoints) {
                            ArrayList<Double> finalCoordinates =
isochrone.applyElevationsToPoints(coordinates, elevationAtPoints, speedInputValue);
                            getActivity().runOnUiThread(() -> {
                                // PLOT ELEVATION SEARCH AREA
```

```
                                List<List<Point>> POINTS_ELEVATION = new
ArrayList<>();
                                List<Point> OUTER_POINTS_ELEVATION = new
ArrayList<>();

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(1),
finalCoordinates.get(0)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(3),
finalCoordinates.get(2)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(5),
finalCoordinates.get(4)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(7),
finalCoordinates.get(6)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(9),
finalCoordinates.get(8)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(11),
finalCoordinates.get(10)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(13),
finalCoordinates.get(12)));

OUTER_POINTS_ELEVATION.add(Point.fromLngLat(finalCoordinates.get(15),
finalCoordinates.get(14)));
                                POINTS_ELEVATION.add(OUTER_POINTS_ELEVATION);
                                style.addSource(new GeoJsonSource("35",
Polygon.fromLngLats(POINTS_ELEVATION)));
                                style.addLayerBelow(new FillLayer("13",
"35").withProperties(
                                        fillColor(Color.parseColor("#ED3742")),
fillOpacity(0.6f)), "style"
                                ); // Plot the polygon
                                LatLngBounds latLngBounds = new
LatLngBounds.Builder()
                                        .include(new LatLng(finalCoordinates.get(0),
finalCoordinates.get(1)))
                                        .include(new LatLng(finalCoordinates.get(2),
finalCoordinates.get(3)))
                                        .include(new LatLng(finalCoordinates.get(4),
finalCoordinates.get(5)))
                                        .include(new LatLng(finalCoordinates.get(6),
finalCoordinates.get(7)))
                                        .include(new LatLng(finalCoordinates.get(8),
finalCoordinates.get(9)))
                                        .include(new
LatLng(finalCoordinates.get(10), finalCoordinates.get(11)))
                                        .include(new
LatLng(finalCoordinates.get(12), finalCoordinates.get(13)))
                                        .include(new
LatLng(finalCoordinates.get(14), finalCoordinates.get(15)))
```

```
                                        .build();

mapboxMap.animateCamera(CameraUpdateFactory.newLatLngBounds(latLngBounds, 100),
3000); // Make the camera zoom over the area of the isochrone
                                FloatingActionButton floatingActionButton =
mapView.findViewById(R.id.clearFloatingActionButton);
                                floatingActionButton.show();
                                floatingActionButton.setOnClickListener(new
View.OnClickListener() { // Create a button nd set the listener to allow the user
to clear the map
                                    @Override
                                    public void onClick(View v) {
                                        MapFragment mapFragment = new MapFragment();

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
                                    }
                                });
                            });
                        }
                    });
                }
            }
        });
        mapboxMap.addOnMapClickListener(this); // Adds an on click listener to the
map
    }

    @Override
    public boolean onMapClick(@NonNull LatLng point) {
        // This runs whenever the map is "tapped"/clicked - This clears any markers
present
        if(markerView != null) {
            markerViewManager.removeMarker(markerView);
            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            markerView = new MarkerView(point, customView);
            markerViewManager.addMarker(markerView);
            setStartLat(point.getLatitude());
            setStartLon(point.getLongitude());
            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
            view.setVisibility(View.VISIBLE);
        } else {
            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            markerView = new MarkerView(point, customView);
            markerViewManager.addMarker(markerView);
```

```
            setStartLat(point.getLatitude());
            setStartLon(point.getLongitude());
            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
            view.setVisibility(View.VISIBLE);
        }
        return true;
    }

    @SuppressWarnings({"MissingPermission"})
    private void enableLocation(@NonNull Style mapStyle) {
        if(ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
            activity = (MainActivity) getContext(); //Essentially replacing "this"
as "this" can't be used in fragments as there is no context passed here
            LocationComponent locationComponent = mapboxMap.getLocationComponent();
            LocationComponentActivationOptions locationComponentActivationOptions =
LocationComponentActivationOptions.builder(activity,
mapStyle).useDefaultLocationEngine(false).build();

locationComponent.activateLocationComponent(locationComponentActivationOptions);
            locationComponent.setLocationComponentEnabled(true);
            locationComponent.setCameraMode(CameraMode.TRACKING);
            //locationComponent.setRenderMode(RenderMode.COMPASS); Removed due to a
known issue with the library here:
https://github.com/mapbox/mapbox-gl-native/issues/14889 - Reported fixed
https://github.com/mapbox/mapbox-gl-native-android/pull/19 with build 8.5.0-beta.1
but error still occurring so it has been commented out

            initLocationEngine();
        } else {
            requestPermissions(
                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    1
            ); //Request location permission if not already granted
        }
    }

    @SuppressWarnings({"MissingPermission"})
    private void initLocationEngine() {
        locationEngine = LocationEngineProvider.getBestLocationEngine(activity);
//Initialise new location engine
        LocationEngineRequest request = new
LocationEngineRequest.Builder(DEFAULT_INTERVAL)
                .setPriority(LocationEngineRequest.PRIORITY_HIGH_ACCURACY)
                .setMaxWaitTime(DEFAULT_MAX_TIME).build(); //Set time intervals for
updating user location (2s)
        locationEngine.requestLocationUpdates(request, callback,
Looper.getMainLooper());
        locationEngine.getLastLocation(callback); //Pass it to callback (ie
LocationChangeListeningActivityLocationCallback)
    }

    @Override
```

```
   public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
       //Method called when requestPermissions(); gets called
       if(requestCode == 1) {
           if (permissions[0].equals(Manifest.permission.ACCESS_FINE_LOCATION) &&
grantResults[0] == PackageManager.PERMISSION_GRANTED) {
               //Check if location permissions have been granted
               mapboxMap.getStyle(new Style.OnStyleLoaded() {
                   @Override
                   public void onStyleLoaded(@NonNull Style style) {
                       enableLocation(style); //Rerun method, this time with
correct permissions
                   }
               });
           }
       } else {
           Toast.makeText(getActivity(), R.string.location_access_not_permitted,
Toast.LENGTH_LONG).show(); //Let user know they need to allow location permission
       }
   }

   private void addressToCoordinates(@NonNull String address) {
       address = address.replaceAll("/[^a-zA-Z0-9 ]/g", "").replaceAll(" ", "%20");
       String url = "https://api.mapbox.com/geocoding/v5/mapbox.places/" + address
+ ".json?&access_token=" + getString(R.string.mapbox_token);
       //Variables must be declared before starting a thread as they have to be
final or effectively-final
       //Creating a thread - running network operations must happen on another
thread as the main/UI thread can skip frames or lifecycle methods
       Thread thread = new Thread(() -> {
           Thread.currentThread().setPriority(Thread.MIN_PRIORITY); //Important to
set the thread priority to MIN or at least less than main/UI thread to prevent
skipping frames or lifecycle methods
           OkHttpClient okHttpClient = new OkHttpClient(); //Initialising an
instance of the HTTP client
           Request request = new Request.Builder() //Build request with URL and
optional headers (not needed)
                   .url(url)
                   .build();
           Response response = null; //Cannot run query in a try(query here){} due
to a target API mismatch (so it runs on more devices)
           try {
               response = okHttpClient.newCall(request).execute();
               String responseStr = response.body().string();
               JSONObject jsonObject = new JSONObject(responseStr); //Convert the
string into a JSONObject for manipulation and data reading
               JSONArray allFeatures = jsonObject.getJSONArray("features");
               JSONObject firstObject = allFeatures.getJSONObject(0);
               JSONObject geometryObject = firstObject.getJSONObject("geometry");
               JSONArray coordPair = geometryObject.getJSONArray("coordinates");
//Final coordinate pair taken from API response in order to plot marker on the map
               double lat = ((Number) coordPair.get(1)).doubleValue();
               double lng = ((Number) coordPair.get(0)).doubleValue();
               setStartLat(lat);
```

```
                setStartLon(lng);
                getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if(markerView != null) {
                            markerViewManager.removeMarker(markerView);
                            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
                            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                            markerView = new MarkerView(new LatLng(lat, lng),
customView);
                            markerViewManager.addMarker(markerView);
                            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
                            view.setVisibility(View.VISIBLE);
                        } else {
                            View customView =
LayoutInflater.from(getContext()).inflate(R.layout.location_marker_holder, null);
                            customView.setLayoutParams(new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                            markerView = new MarkerView(new LatLng(lat, lng),
customView);
                            markerViewManager.addMarker(markerView);
                            ViewGroup view = (ViewGroup)
getActivity().findViewById(R.id.startSearchRectangle);
                            view.setVisibility(View.VISIBLE);
                        }
                    }
                });
            } catch (JSONException e) {
                Toast.makeText(getActivity(), "Failed to plot point\nPlease try
again later", Toast.LENGTH_LONG).show();
            } catch(IOException e) {
                Toast.makeText(getActivity(), R.string.ioexception_error_message,
Toast.LENGTH_LONG).show();
            } finally {
                if(response != null) {
                    response.body().close(); //Close the client after checking it
existed in the first place
                }
            }
        });
        thread.start();
    }

    private void clearScreen() {
        // Remove all markers and the start search rectangle
        View view = (View) getActivity().findViewById(R.id.startSearchRectangle);
        if (view.getVisibility()== View.VISIBLE) {
            view.setVisibility(View.GONE);
        }
```

```
        markerViewManager.removeMarker(markerView);
        final EditText mapSearchBox = (EditText)
getActivity().findViewById(R.id.map_search_bar);
        mapSearchBox.getText().clear();
    }

    private void showSearchInput() {
        // Brings up new search screen
        View view = (View) getActivity().findViewById(R.id.startSearchRectangle);
        if (view.getVisibility() == View.VISIBLE) {
            view.setVisibility(View.GONE);
        }
        Fragment newSearchFragment = new NewSearchFragment();
        Bundle bundle = new Bundle();
        bundle.putDouble("lat", startLat);
        bundle.putDouble("lon", startLon);
        newSearchFragment.setArguments(bundle);
        getFragmentManager().beginTransaction().replace(R.id.fragment_container,
newSearchFragment).commit();
    }

    private static class LocationChangeListeningActivityLocationCallback implements
LocationEngineCallback<LocationEngineResult> {

        private final WeakReference<MapFragment> fragmentWeakReference;

        LocationChangeListeningActivityLocationCallback(MapFragment fragment) {
            this.fragmentWeakReference = new WeakReference<>(fragment);
        }

        @Override
        public void onSuccess(LocationEngineResult result) {
            MapFragment fragment = fragmentWeakReference.get();

            if(fragment != null) {
                Location location = result.getLastLocation(); //Fetches last
location

                if(location == null) {
                    return;
                }
                if(fragment.mapboxMap != null && result.getLastLocation() != null) {

fragment.mapboxMap.getLocationComponent().forceLocationUpdate(result.getLastLocatio
n()); //Forces map to update user location to last lat lon from location engine
                }
            }
        }

        @Override
        public void onFailure(@NonNull Exception ignored) {

        }
```

```
}

private double getStartLat() {
    return this.startLat;
}

private void setStartLat(double lat) {
    this.startLat = lat;
}

private double getStartLon() {
    return this.startLon;
}

private void setStartLon(double lon) {
    this.startLon = lon;
}

@SuppressWarnings({"MissingPermission"})
@Override
public void onStart() {
    super.onStart();
    mapView.onStart();
}

@Override
public void onSaveInstanceState(@NonNull Bundle outState) {
    super.onSaveInstanceState(outState);
    mapView.onSaveInstanceState(outState);
}

@Override
public void onDestroyView() {
    super.onDestroyView();
    if(locationEngine != null) {
        locationEngine.removeLocationUpdates(callback);
    }
    if(mapView != null) {
        mapView.onDestroy();
    }
}

@Override
public void onPause() {
    super.onPause();
    mapView.onPause();
}

@Override
public void onResume() {
    super.onResume();
    mapView.onResume();
}
```

```
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <FrameLayout
       android:id="@+id/fragment_container"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:layout_above="@id/bottom_navigation"/>

   <com.google.android.material.bottomnavigation.BottomNavigationView
       android:id="@+id/bottom_navigation"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:layout_alignParentBottom="true"
       android:layout_marginBottom="0dp"
       android:background="?android:attr/windowBackground"
       app:menu="@menu/bottom_navigation" />

</RelativeLayout>
```

location_marker_holder.xml (used for the location icon on the map)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:map="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="horizontal">

   <ImageView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:contentDescription="@string/app_name"
       android:src="@drawable/location_marker"/>

</RelativeLayout>
```

SearchesFragment.java

```
package com.bengavin.missingpersons;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
```

```
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

public class SearchesFragment extends Fragment {

   private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
   RecyclerView recyclerView;
   CardAdapter cardAdapter;

   @Nullable
   @Override
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       View fragmentView = inflater.inflate(R.layout.fragment_searches, container,
false); // Inflate the layout so it can be seen in the activity
       return fragmentView;
   }

   @Override
   public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
       super.onViewCreated(view, savedInstanceState);

       recyclerView = view.findViewById(R.id.recyclerView);
       recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

       getSearches(); // This function fetches all of the searches from the online
database
   }

   private void getSearches() {
       onlineDatabase.collection("searches")
               .get()
               .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() { //
Make the call for all searches to the database, adds a listener that fires when the
data comes back as it takes x amount of time
                   @Override
                   public void onComplete(@NonNull Task<QuerySnapshot> task) {
                       ArrayList<CardModel> cards = new ArrayList<>();
                       if(task.isSuccessful()) {
                           for(QueryDocumentSnapshot documentSnapshot :
task.getResult()) { // This loops through the data and creates a new instance of
CardModel which will create a CardView
                               CardModel cardModel = new CardModel();
```

```
                                // Use all the setters and getters to show the
relevant information on each card

cardModel.setSearchName(documentSnapshot.getString("searchName"));
                                cardModel.setSearchOwner("Owned by: " +
documentSnapshot.getString("owner"));
                                cardModel.setUid(documentSnapshot.getId());
                                cards.add(cardModel);
                            }
                            cardAdapter = new CardAdapter(getActivity(), cards);
                            recyclerView.setAdapter(cardAdapter); // Setting the
adapter makes the compiled CardView list show up
                        } else {
                            Toast.makeText(getContext(), "Failed to load
searches\nPlease try again", Toast.LENGTH_LONG).show(); // Show an error message if
it fails to fetch the searches
                        }
                    }
                });
    }

}
```

searches_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

search_card.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardCornerRadius="5dp"
    android:elevation="3dp"
    app:cardUseCompatPadding="true">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">

        <TextView
```

```
            android:id="@+id/searchName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/search_name_placeholder"
            android:textColor="#000"
            android:textSize="20sp"/>

        <TextView
            android:id="@+id/searchOwner"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/search_owner_placeholder"
            android:layout_marginTop="25dp"/>

    </RelativeLayout>

</androidx.cardview.widget.CardView>
```

CardAdapter.java

```
package com.bengavin.missingpersons;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.recyclerview.widget.RecyclerView;

public class CardAdapter extends RecyclerView.Adapter<CardHolder> {

    Context context;
    ArrayList<CardModel> cardModels;

    public CardAdapter(Context context, ArrayList<CardModel> cardModels) {
        this.context = context;
        this.cardModels = cardModels;
    }

    @NonNull
    @Override
    public CardHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.search_card, null);
//Inflate search_card.xml when the view holder gets created

        return new CardHolder(view);
    }
```

```
    @Override
    public void onBindViewHolder(@NonNull CardHolder holder, int position) {
        holder.mSearchName.setText(cardModels.get(position).getSearchName());
        holder.mSearchOwner.setText(cardModels.get(position).getSearchOwner());

        holder.setCardClickListener(new CardClickListener() {
            //Use interface to make a click listener
            @Override
            public void onCardClickListener(View view, int position) {
                String name = cardModels.get(position).getSearchName();
                String owner =
cardModels.get(position).getSearchOwner().substring(10);
                String uid = cardModels.get(position).getUid();
                //Get attributes from the clicked search card

                Fragment fragment = new SearchInsightFragment();
                Bundle bundle = new Bundle();
                bundle.putString("name", name);
                bundle.putString("owner", owner);
                bundle.putString("uid", uid);
                fragment.setArguments(bundle);
                //This makes a new fragment and puts the attributes retrieved above,
into a bundle which can be used by the secondary fragment

                FragmentManager fragmentManager =
((AppCompatActivity)context).getSupportFragmentManager();
//((AppCompatActivity)context) used for context due this extending an Adapter
                fragmentManager.beginTransaction().replace(R.id.fragment_container,
fragment).addToBackStack(null).commit(); //Change fragments
            }
        });
    }

    @Override
    public int getItemCount() {
        return cardModels.size();
    }
}
```

CardHolder.java

```
package com.bengavin.missingpersons;

import android.view.View;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class CardHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

    TextView mSearchName, mSearchOwner;
    CardClickListener cardClickListener;
```

```
    CardHolder(@NonNull View itemView) {
        super(itemView);

        this.mSearchName = itemView.findViewById(R.id.searchName);
        this.mSearchOwner = itemView.findViewById(R.id.searchOwner);

        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        this.cardClickListener.onCardClickListener(v, getLayoutPosition());
    }

    public void setCardClickListener(CardClickListener cardClickListener) {
        this.cardClickListener = cardClickListener;
    }

}
```

CardModel.java

```
package com.bengavin.missingpersons;

public class CardModel {

    //This class just contains getters and setters for the card model

    private String searchName, searchOwner, uid;

    public String getSearchName() {
        return searchName;
    }

    public void setSearchName(String searchName) {
        this.searchName = searchName;
    }

    public String getSearchOwner() {
        return searchOwner;
    }

    public void setSearchOwner(String searchOwner) {
        this.searchOwner = searchOwner;
    }

    public String getUid() {
        return uid;
    }

    public void setUid(String uid) {
        this.uid = uid;
    }
```

```
}
```

CardClickListener.java

```
package com.bengavin.missingpersons;

import android.view.View;

public interface CardClickListener {

   void onCardClickListener(View view, int position);

}
```

SearchInsightFragment.java

```
package com.bengavin.missingpersons;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.provider.Settings;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class SearchInsightFragment extends Fragment {

   private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
   private Button backButton, openMapButton;
   private TextView mSearchName, mSearchOwner, mMissingPersonName, mSpeed,
mStartLatLon, mStartTime;
   private float speed;
   private double startLat, startLon;
   private long time;

   @Nullable
   @Override
```

```
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       //When the fragment gets created this will inflate the fragment view
       View fragmentView = inflater.inflate(R.layout.fragment_search_insight,
container, false);
       return fragmentView;
   }

   @SuppressWarnings({"HardwareIds"})
   @Override
   public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
       super.onViewCreated(view, savedInstanceState);

       backButton = view.findViewById(R.id.insightBackButton);
       openMapButton = view.findViewById(R.id.insightOpenMapButton);

       mSearchName = view.findViewById(R.id.insightSearchName);
       mSearchOwner = view.findViewById(R.id.insightSearchOwner);
       mMissingPersonName = view.findViewById(R.id.insightMissingPersonName);
       mSpeed = view.findViewById(R.id.insightSpeed);
       mStartLatLon = view.findViewById(R.id.insightLatLon);
       mStartTime = view.findViewById(R.id.insightStartTime);

       backButton.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               getFragmentManager().popBackStackImmediate();
           }
       });

       openMapButton.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               MapFragment mapFragment = new MapFragment();
               Bundle newBundle = new Bundle();
               newBundle.putBoolean("polygonToPlot", true);
               newBundle.putFloat("speedInputValue", speed);
               newBundle.putLong("startTime", time);
               newBundle.putDouble("lat", startLat);
               newBundle.putDouble("lon", startLon);
               mapFragment.setArguments(newBundle);

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
           }
       });

       Bundle bundle = this.getArguments(); //Get the arguments passed by
CardAdapter from the bundle
       String uid = bundle.getString("uid", "NO_KEY");
       if (uid.equals("NO_KEY")) {
           Toast.makeText(getActivity(), "An error occurred whilst fetching the
search\nPlease try again later.", Toast.LENGTH_LONG).show();
       } else {
```

```
            fetchData(uid, new FirestoreDataCallback() {
                @Override
                public void onData(String searchName, String owner, String ownerID,
String missingPersonName, String speed, String startLat, String startLon, String
startTime, String documentUID) {
                    setSpeed(Float.valueOf(speed));
                    setTime(Long.valueOf(startTime));
                    setStartLat(Double.valueOf(startLat));
                    setStartLon(Double.valueOf(startLon));
                    Date date = new Date(Long.valueOf(startTime));
                    // Put time back into a readable format for the user
                    SimpleDateFormat simpleDateFormat = new SimpleDateFormat("HH:mm
dd-MM-yyyy z", Locale.getDefault());
                    String dateString = simpleDateFormat.format(date);
                    mSearchName.setText(searchName);
                    mSearchOwner.setText(owner);
                    mMissingPersonName.setText(missingPersonName);

mSpeed.setText(String.valueOf(BigDecimal.valueOf(Float.valueOf(speed)).setScale(3,
RoundingMode.HALF_UP).floatValue()));
                    mStartLatLon.setText(getString(R.string.lat_lon_template,
startLat, startLon));
                    mStartTime.setText(dateString);

                    String androidUID =
Settings.Secure.getString(getContext().getContentResolver(),
Settings.Secure.ANDROID_ID);
                    if(androidUID.equals(ownerID)) {
                        Button deleteButton =
view.findViewById(R.id.insightDeleteButton);
                        deleteButton.setVisibility(View.VISIBLE);
                        deleteButton.setOnClickListener(new View.OnClickListener() {
                            @Override
                            public void onClick(View v) {
                                new AlertDialog.Builder(getContext())
                                        .setTitle("Confirm action")
                                        .setMessage("Are you sure you want to delete
this search? This action cannot be undone.")
                                        .setIcon(R.drawable.ic_error_black_24dp)
                                        .setPositiveButton(R.string.yes, new
DialogInterface.OnClickListener() {
                                            @Override
                                            public void onClick(DialogInterface
dialog, int which) {

onlineDatabase.collection("searches")
                                                    .document(documentUID)
                                                    .delete()
                                                    .addOnSuccessListener(new
OnSuccessListener<Void>() {
                                                        @Override
                                                        public void
onSuccess(Void aVoid) {
```

```
Toast.makeText(getActivity(), "Successfully deleted search",
Toast.LENGTH_SHORT).show();

getFragmentManager().popBackStackImmediate();
                                                                    }
                                                                })
                                                                .addOnFailureListener(new
OnFailureListener() {
                                                                    @Override
                                                                    public void
onFailure(@NonNull Exception e) {

Toast.makeText(getActivity(), "Failed to delete search\nPlease try again later",
Toast.LENGTH_SHORT).show();
                                                                    }
                                                                });
                                                    }
                                                })
                                                .setNegativeButton(R.string.no, null)
                                                .show();
                                        }
                                    });
                                }
                            }
                        });
                }
            }

    private void fetchData(String uid, FirestoreDataCallback firestoreDataCallback)
{
        onlineDatabase.collection("searches")
                .document(uid)
                .get()
                .addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                    @Override
                    public void onSuccess(DocumentSnapshot documentSnapshot) {
                        String searchName =
documentSnapshot.getString("searchName");
                        String owner = documentSnapshot.getString("owner");
                        String ownerID = documentSnapshot.getString("ownerID");
                        String missingPersonName =
documentSnapshot.getString("missingPersonName");
                        String speed =
String.valueOf(documentSnapshot.get("speed"));
                        String startLat =
String.valueOf(documentSnapshot.get("startLat"));
                        String startLon =
String.valueOf(documentSnapshot.get("startLon"));
                        String startTime =
String.valueOf(documentSnapshot.get("startTime"));
                        String documentUID = documentSnapshot.getId();
                        firestoreDataCallback.onData(searchName, owner, ownerID,
missingPersonName, speed, startLat, startLon, startTime, documentUID);
```

```
                }
            });
    }

    public double getSpeed() {
        return speed;
    }

    public void setSpeed(float speed) {
        this.speed = speed;
    }

    public double getStartLat() {
        return startLat;
    }

    public void setStartLat(double startLat) {
        this.startLat = startLat;
    }

    public double getStartLon() {
        return startLon;
    }

    public void setStartLon(double startLon) {
        this.startLon = startLon;
    }

    public long getTime() {
        return time;
    }

    public void setTime(long time) {
        this.time = time;
    }
}
```

fragment_search_insight.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="15dp">

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
```

```
        android:layout_weight="0.5"
        android:text="@string/search_name"
        android:textSize="20sp"/>

    <TextView
        android:id="@+id/insightSearchName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:textSize="20sp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="15dp">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:text="@string/missing_person_name"
        android:textSize="20sp"/>

    <TextView
        android:id="@+id/insightMissingPersonName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:textSize="20sp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="15dp">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:text="@string/search_owner_name"
        android:textSize="20sp"/>

    <TextView
        android:id="@+id/insightSearchOwner"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:textSize="20sp"/>
```

```
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="15dp">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:text="@string/speed"
        android:textSize="20sp"/>

    <TextView
        android:id="@+id/insightSpeed"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:textSize="20sp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="15dp">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:text="@string/search_start_date_time"
        android:textSize="20sp"/>

    <TextView
        android:id="@+id/insightStartTime"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:textSize="20sp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="15dp">

    <TextView
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:text="@string/lat_lon"
        android:textSize="20sp"/>

    <TextView
        android:id="@+id/insightLatLon"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:textSize="20sp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp">

    <Button
        android:id="@+id/insightBackButton"
        android:text="@string/back"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"/>

    <Button
        android:id="@+id/insightOpenMapButton"
        android:text="@string/open_map"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:backgroundTint="@color/colorPrimary"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:gravity="center">

    <Button
        android:id="@+id/insightDeleteButton"
        android:text="@string/delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawableLeft="@drawable/ic_delete_black_24dp"
        android:drawablePadding="7.5dp"
        android:visibility="gone"/>

</LinearLayout>
```

```
</LinearLayout>
```

NewSearchFragment.java

```
package com.bengavin.missingpersons;

import android.app.AlertDialog;
import android.os.Bundle;
import android.provider.Settings;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.mapbox.mapboxsdk.geometry.LatLng;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class NewSearchFragment extends Fragment {

   private FirebaseFirestore onlineDatabase = FirebaseFirestore.getInstance();
   private Button cancelButton, submitButton, setDateTimeButton;
   private EditText searchName, speedInput, missingPersonName, ownerName;
   private TextView timeOutput;
   private Bundle bundle;
   private long time = 0;
   private double lat, lon;

   @Nullable
   @Override
   public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
       View fragmentView = inflater.inflate(R.layout.fragment_new_search,
container, false);
       return fragmentView;
   }
```

```
    @Override
    public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        setDateTimeButton = view.findViewById(R.id.set_date_time);
        cancelButton = view.findViewById(R.id.cancel_button);
        submitButton = view.findViewById(R.id.submit_button);
        timeOutput = view.findViewById(R.id.time_picker_output);

        bundle = this.getArguments();
        lat = bundle.getDouble("lat", 0);
        lon = bundle.getDouble("lon", 0);

        setDateTimeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final View dialogView = View.inflate(getActivity(),
R.layout.date_time_picker, null);
                final AlertDialog alertDialog = new
AlertDialog.Builder(getActivity()).create(); // This builds an alert dialog which
is basically a popup

                dialogView.findViewById(R.id.set).setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View inner) {
                        DatePicker datePicker =
alertDialog.findViewById(R.id.date_picker);
                        TimePicker timePicker =
alertDialog.findViewById(R.id.time_picker);

                        timePicker.setIs24HourView(true);

                        // Get a readable date format
                        Calendar calendar = new
GregorianCalendar(datePicker.getYear(), datePicker.getMonth(),
datePicker.getDayOfMonth(), timePicker.getCurrentHour(),
timePicker.getCurrentMinute());

                        time = calendar.getTimeInMillis();

timeOutput.setText(getString(R.string.date_time_output_template,
String.valueOf(timePicker.getCurrentHour()),
String.valueOf(timePicker.getCurrentMinute()),
String.valueOf(datePicker.getDayOfMonth()), String.valueOf(datePicker.getMonth()),
String.valueOf(datePicker.getYear())));
                        // This shows the user the time they selected on the new
search screen to confirm that they selected the date
                        alertDialog.dismiss();
                    }
                });
                alertDialog.setView(dialogView);
                alertDialog.show();
            }
```

```
        });

        cancelButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

getFragmentManager().beginTransaction().replace(R.id.fragment_container, new
MapFragment()).commit();
            }
        });

        submitButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                searchName = view.findViewById(R.id.search_name_input);
                String searchNameInput = searchName.getText().toString();
                speedInput = view.findViewById(R.id.speed_input);
                missingPersonName =
view.findViewById(R.id.missing_person_name_input);
                String missingPersonNameInput =
missingPersonName.getText().toString();
                ownerName = view.findViewById(R.id.search_owner_input);
                String ownerNameInput = ownerName.getText().toString();
                float speedInputValue = 0;
                try {
                    speedInputValue =
Float.parseFloat(speedInput.getText().toString());
                    if(searchNameInput.length() == 0 ||
missingPersonNameInput.length() == 0 || ownerNameInput.length() == 0) {
                        Toast.makeText(getActivity(), "Please fill in all of the
fields", Toast.LENGTH_SHORT).show();
                        if (time == 0) {
                            time = System.currentTimeMillis();
                        }
                    } else {
                        createSearch(searchNameInput, ownerNameInput,
bundle.getDouble("lat", 0), bundle.getDouble("lon", 0), speedInputValue,
missingPersonNameInput, time);
                    }
                } catch (NumberFormatException e) {
                    Toast.makeText(getActivity(), "Please enter a speed",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private void createSearch(String searchName, String ownerName, double startLat,
double startLon, float speed, String missingPersonName, long time) {
        // This function puts all of the parameters into a Map and then sends it to
the online database
        String androidUID =
Settings.Secure.getString(getContext().getContentResolver(),
Settings.Secure.ANDROID_ID);
```

```
        Map<String, Object> data = new HashMap<>();
        data.put("searchName", searchName);
        data.put("owner", ownerName);
        data.put("ownerID", androidUID);
        data.put("startLat", startLat);
        data.put("startLon", startLon);
        data.put("speed", speed);
        data.put("missingPersonName", missingPersonName);
        data.put("startTime", time);

        onlineDatabase.collection("searches")
                .add(data)
                .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                    @Override
                    public void onSuccess(DocumentReference documentReference) {
                        Toast.makeText(getActivity(), "Search created and saved to
online database", Toast.LENGTH_SHORT).show();
                        MapFragment mapFragment = new MapFragment(); // Switch back
to the map
                        Bundle newBundle = new Bundle();
                        newBundle.putBoolean("polygonToPlot", true); // Plot the new
search that was just created
                        newBundle.putFloat("speedInputValue", speed);
                        newBundle.putLong("startTime", time);
                        newBundle.putDouble("lat", lat);
                        newBundle.putDouble("lon", lon);
                        mapFragment.setArguments(newBundle);

getFragmentManager().beginTransaction().replace(R.id.fragment_container,
mapFragment).commit();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(getActivity(), "Search created but failed to
save to online database", Toast.LENGTH_SHORT).show(); // Alert the user something
went wrong saving the search
                    }
                });
    }

}
```

fragment_new_search.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="20dp"
    android:orientation="horizontal" >

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:textSize="15sp"
        android:text="@string/search_name" />

    <EditText
        android:id="@+id/search_name_input"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:inputType="text"
        android:ems="10" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="20dp"
    android:orientation="horizontal" >

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:textSize="15sp"
        android:text="@string/speed" />

    <EditText
        android:id="@+id/speed_input"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:inputType="numberDecimal"
        android:ems="10" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="20dp"
    android:orientation="horizontal" >
```

```
    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:textSize="15sp"
        android:text="@string/set_search_start_date_time" />

    <Button
        android:id="@+id/set_date_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/search_start_date_time"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="end">

    <TextView
        android:id="@+id/time_picker_output"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="end"
        android:layout_marginRight="30dp"
        android:layout_marginEnd="30dp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="20dp"
    android:orientation="horizontal" >

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:textSize="15sp"
        android:text="@string/missing_person_name" />

    <EditText
        android:id="@+id/missing_person_name_input"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:inputType="text"
        android:ems="10" />
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="20dp"
    android:orientation="horizontal" >

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:textSize="15sp"
        android:text="@string/search_owner_name" />

    <EditText
        android:id="@+id/search_owner_input"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.50"
        android:inputType="text"
        android:ems="10" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <Button
        android:id="@+id/cancel_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginStart="20dp"
        android:layout_marginRight="10dp"
        android:layout_marginEnd="10dp"
        android:layout_weight="0.50"
        android:ems="5"
        android:text="@string/cancel" />

    <Button
        android:id="@+id/submit_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginStart="10dp"
        android:layout_marginRight="20dp"
        android:layout_marginEnd="20dp"
        android:layout_weight="0.50"
        android:ems="5"
        android:backgroundTint="@color/colorPrimary"
        android:text="@string/create" />
```

```
    </LinearLayout>

</LinearLayout>
```

date_time_picker.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:padding="8dp"
    android:layout_height="match_parent">

    <DatePicker
        android:id="@+id/date_picker"
        android:layout_width="match_parent"
        android:calendarViewShown="true"
        android:spinnersShown="false"
        android:layout_weight="4"
        android:datePickerMode="calendar"
        android:layout_height="0dp" />

    <TimePicker
        android:id="@+id/time_picker"
        android:layout_weight="4"
        android:layout_width="match_parent"
        android:timePickerMode="clock"
        android:layout_height="0dp" />

    <Button
        android:id="@+id/set"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:text="Set"
        android:layout_height="0dp"
        android:backgroundTint="@color/colorPrimary"/>

</LinearLayout>
```

Isochrone.java

```
package com.bengavin.missingpersons;

import android.content.Context;

import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.NoSuchElementException;
```

```
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class Isochrone {

   private Context context;
   private volatile ArrayList<Integer> elevationsAtPoints = new ArrayList<>();
   private int HEIGHT_PER_MINUTE = 10; // For every 10 metres travelled in
elevation  ≈ 1 minute of travel

   public Isochrone(Context context) {
       this.context = context;
   }

   private double toRadians(double degrees) {
       double radians = degrees * ((Math.PI)/180);
       return radians;
   }

   private double toDegrees(double radians) {
       double factor = 180 / Math.PI;
       return radians * factor;
   }

   public ArrayList<Double> getMaxCoordinates(double[] startingPair, double
distance) { // This method will calculate the maximum distance covered at a
constant speed
       double lat = toRadians(startingPair[0]);
       double lon = toRadians(startingPair[1]);
       double distanceRatio = distance/6371.01; //Divide the distance given by the
radius of the Earth to get a ratio
       ArrayList<Double> endMaxCoordinates = new ArrayList<Double>(); //Initialise
end array for all coordinates (size 16 as 8 pairs)
       for(int i = 0; i < 360; i+= 45) { // <360 as 360 should not be included as
it is the same as 0
           double bearing = toRadians(i);
           double endLat = Math.asin(Math.sin(lat)*Math.cos(distanceRatio) +
Math.cos(lat)*Math.sin(distanceRatio)*Math.cos(bearing)); // Calculate the new
latitude using trigonometry
           endMaxCoordinates.add(toDegrees(endLat));
           double endLon = lon +
Math.atan2(Math.sin(bearing)*Math.sin(distanceRatio)*Math.cos(lat),
Math.cos(distanceRatio)-Math.sin(lat)*Math.sin(endLat)); // Calculate the new
latitude using trigonometry
           endMaxCoordinates.add(toDegrees(endLon));
       }
       return endMaxCoordinates; // Return an arraylist that contains all of the
new coordinates
   }

   public long getTimeDifference(long startTime) { // This method is used to allow
the search area to self expand as time passes
```

```
        startTime = startTime/1000L;
        long unixTime = System.currentTimeMillis()/1000L;
        long difference = unixTime - startTime;
        return difference;
    }

    public double getDistanceTravelled(double timeDifference, double speed) {
        timeDifference = timeDifference/60/60;  // Gets time in hours
        double displacement = timeDifference * speed;
        return displacement; // Displace in km/h
    }

    public void getElevations(ArrayList<Double> coordinates, TilequeryCallback
tilequeryCallback) { // This method is responsible for making the API request to an
endpoint that can return topography data about the given point
        Thread thread = new Thread(() -> {
            int j = 0;
            int k = 1;
            for (int i = 0; i < 9; i++) {
                // Loop through the number of points
                String url =
"https://api.mapbox.com/v4/mapbox.mapbox-terrain-v2/tilequery/" +
coordinates.get(k) + "," + coordinates.get(j) +
".json?layers=contour&limit=50&access_token=" +
context.getString(R.string.mapbox_token);
                Thread.currentThread().setPriority(Thread.MIN_PRIORITY);
                OkHttpClient okHttpClient = new OkHttpClient();
                Request request = new Request.Builder()
                        .url(url)
                        .build();
                Response response = null;
                // Build an HTTP GET request
                try {
                    response = okHttpClient.newCall(request).execute();
                    String responseStr = response.body().string();
                    // API returns a JSON response which needs to be parsed
                    JSONObject jsonObject = new JSONObject(responseStr);
                    JSONArray allFeatures = jsonObject.getJSONArray("features");
                    ArrayList<Integer> elevations = new ArrayList<>();
                    for (int m = 0; m < allFeatures.length(); m++) {
                        // Features need to be looped through in order to find the
highest point because different spots on the same point can have different features
so the highest one is used as a guide
                        JSONObject feature = allFeatures.getJSONObject(m);
                        JSONObject properties = feature.getJSONObject("properties");
                        int featureElevation = (Integer) properties.get("ele"); //
API always returns an int
                        elevations.add(featureElevation);
                    }
                    int highestElevation = 0;
                    try {
                        highestElevation = Collections.max(elevations); // Get max
elevation
                    } catch (NoSuchElementException e) {
```

```
                    // Alert the user which point failed to be adjusted
according to height
                    Toast.makeText(context, "The elevation for point " + i + "
failed to load - using max possible distance", Toast.LENGTH_SHORT).show();
                }
                elevationsAtPoints.add(highestElevation);
            } catch (JSONException e) {
                // If this is thrown, something likely went wrong with the API
so the user will need to try again later
                Toast.makeText(context, "Failed to plot point\nPlease try again
later", Toast.LENGTH_LONG).show();
            } catch (IOException e) {
                // Alert the user there was a connectivity problem (such as no
internet etc)
                Toast.makeText(context,
context.getString(R.string.ioexception_error_message), Toast.LENGTH_LONG).show();
            } finally {
                if(response != null) {
                    response.body().close();
                }
            }
            j = j + 2;
            k = k + 2;
            if(i == 8) {
                // Run the callback on the final iter
                tilequeryCallback.onData(elevationsAtPoints);
            }
        }
    });
    thread.start();
  }

  private ArrayList<Double> retractDistance(int bearing, double lat, double lon,
double distance) {
    // This method does the same as getMaxCoordinates although it inverts the
bearing in order to make the direction the opposite to bring the max coordinate
back by x amount
    lat = toRadians(lat);
    lon = toRadians(lon);
    double oppositeBearing = toRadians(360 - bearing); // Invert
bearing/direction
    double distanceRatio = distance/6371.01;
    ArrayList<Double> endCoordinates = new ArrayList<>();
    double endLat = Math.asin(Math.sin(lat)*Math.cos(distanceRatio) +
Math.cos(lat)*Math.sin(distanceRatio)*Math.cos(oppositeBearing));
    endCoordinates.add(toDegrees(endLat));
    double endLon = lon +
Math.atan2(Math.sin(oppositeBearing)*Math.sin(distanceRatio)*Math.cos(lat),
Math.cos(distanceRatio)-Math.sin(lat)*Math.sin(endLat));
    endCoordinates.add(toDegrees(endLon));
    return endCoordinates;
  }
```

```
   public ArrayList<Double> applyElevationsToPoints(ArrayList<Double>
pointCoordinates, ArrayList<Integer> elevations, double speed) {
       int j = 0;
       int k = 1;
       int elevationAtStartPoint = elevations.get(elevations.size()-1);
       ArrayList<Double> endCoordinates = new ArrayList<>();
       for(int i = 0; i < pointCoordinates.size()/2; i++) {
           // Loop through the number of points (its a shifted loop because of how
the coordinates are stored in the arraylist
           int bearing = 0;
           double lat = pointCoordinates.get(j);
           double lon = pointCoordinates.get(k);
           int elevationAtPoint = elevations.get(i);
           int heightDifference = elevationAtStartPoint - elevationAtPoint;
           int minutesToAdd = Math.abs(Math.round(heightDifference /
HEIGHT_PER_MINUTE));
           double distanceTravelled = getDistanceTravelled(minutesToAdd * 60,
speed);
           ArrayList<Double> retractionCoordinates = retractDistance(bearing, lat,
lon, distanceTravelled);
           double endLat = retractionCoordinates.get(0);
           endCoordinates.add(endLat);
           double endLon = retractionCoordinates.get(1);
           endCoordinates.add(endLon);
           j = j + 2;
           k = k + 2;
           bearing = bearing + 45;
       }
       return endCoordinates; // Return the final coordinates which can then be
plotted on the map
   }

}
```

FirestoreDataCallback.java

```
package com.bengavin.missingpersons;

public interface FirestoreDataCallback {

   void onData(String searchName, String owner, String ownerID, String
missingPersonName, String speed, String startLat, String startLon, String
startTime, String documentUID);

}
```

TilequeryCallback.java

```
package com.bengavin.missingpersons;

import java.util.ArrayList;

public interface TilequeryCallback {

   void onData(ArrayList<Integer> elevationAtPoints);
```

```
}
```