NYTE SAFE

NyteSafe

A Night-time Safety Application

Group 21

Final Report

Elliot Smith 21075246, Daniel Hixson 21058214

Christopher Petrov 21059629, Dylan Lewis 21090729

Harry Jones 21066051, Wei-Ting (Charlene) Yeh 21058519

Rohin Rashi 21062798, James Riordan 21075030

Rhys Cummins 21090352

Client: Dr Kirill Sidorov

Supervisor: Angga Anggoro

Contents

Introduction

Throughout this report we will cover all the progress made on the project since the previous report in detail, with an appendix for any additional files and documentation as required. We will discuss all the progress made towards creating a final product and make justifications for all changes and decisions we have made throughout the project.

Our Solution

Our solution to the client's problem is a night-time safety app we are calling NyteSafe. This app allows a user to find a route back to their home from any location, whilst monitoring their progress and safety throughout the journey. The app will aim to monitor the users progress along their route and will aim track any deviation from said route, although this could not be integrated as discussed within this report. Should a user deviate too far from their given route the app would have asked them if they are OK, prompting them to enter a PIN, if they fail to respond to this notification then their emergency contact of choice will be notified of their location and that they may be in danger. The app will also monitor the movement of their device, for example if the device moves in an unexpected way the app will decide if this is a normal movement or whether the user may have fallen over or be in some form of confrontation, this will again trigger a notification to their emergency contact. Furthermore, the app will have a heatmap of criminal activity in the area of the user and their route and will try to avoid these areas when planning a route. This will mean the user is taking the safest possible route to get home. Finally, the app would also have offered a facility for emergency medical information to be stored for the event that the user may be incapacitated or otherwise unable to speak, and require medical assistance, allowing first responders to see any allergies or underlying conditions they may have, however this was also not implemented as explained in the report.

Justification Overview

We chose to create a route tracking app as it allowed us to monitor the progress and safety of a user along a route in the most effective, and accurate, way possible. The ability to monitor their deviation, or distance away, from the provided route also allows us the ability to spot any issues in real time by processing the user's current location when compared to the route we have provided for them to follow. Monitoring the user's device movement through accelerometer data enabled us to detect any unexpected events and movement that may occur to the user, for example should they fall over, or end up in an altercation then the app will detect the anomalous movement in real time and help get the user assistance, how we determined anomalous data will be discussed in the report. The use of local crime data allows us to calculate as safe a route as possible. Obviously, this isn't a guarantee that the user will be safe, but it will minimise the risk as much as we can. Finally, we chose to allow the user to store any medical information within the app as it would allow first responders access to any medical history that may be relevant, for example if a user is found unconscious due to a head injury, perhaps from a fall, and the user has an allergy to morphine, the emergency medical information will mean first responders will know not to administer any, despite the user be unable to tell them this themselves.

Solution Completeness

Overall, I believe this report will show how we have made excellent progress towards a complete solution to the problem the client presented to us at the beginning of the project.

There are certainly still some elements of the project that we have not been able to implement that we would have liked to have included, such as being able to detect if the user had unexpectedly entered a vehicle, but this was not achievable in the time available to us. However, we do feel that the features that we have been able to implement do still solve the problem that the client presented to us.

Changes and Client Feedback

Since the Interim Report, we have made various alterations and changes to our project to fit both the feedback received and our own insights, especially those gained by carrying out this project. Firstly, the feedback we have received has been useful to make the workflow more efficient and make the program better. This feedback began with us being told to fully implement the core functions of the application before commencing work on other features. From this, we then began to direct much of our focus to the main functionality of the application, such as anomaly detection and map functions.

Realising that time was a major constraint in this project, this was the right choice, since if we spent more time working on aesthetics and quality of life the core functions would not have been completed. Most of our time was then spent actioning this feedback in the form of implementing these core features.

Another reason to justify the implementation of core features first was that it would enable us to start testing earlier than we had originally expected. We also had to consider the aspect of combining code to create a working application, since multiple people worked on different features, we would need to integrate them all eventually. Completing them in a fast, yet thorough manner would prove to be best.

Another piece of feedback we had received was related to the previous one, it was that we should focus on the quality-of-life changes and aesthetics of the application last. We looked at this in the same light as the previous feedback. This also helped us set the pace for the project, which allowed us to finish the components which were more important as discussed above. Designing aesthetics also proved to be a slow process, with a lot of fine tuning involved. Although, we did not completely stop these developments, we spent some time to create designs that we were happy with and waited until near the end of the project to implement them. These were the reasons that we used to justify leaving a full design overhaul until the end. However, we have had to leave out certain design features such as a bottom-navigation bar, this is unfortunate, but time was very much against us. Though most of the design features that we initially intended on implementing are included in the final version of the application.

The client had also told us that we should think about our testing methods, specifically the beta-testing. From this feedback, we had concluded that it would be best to put most of our effort into in-house testing. The best course of action for us seemed to be to test the application ourselves, which is discussed later in this document. One course of action that we chose to test the application ourselves as it is being developed. This would help speed up the process and give us time to finish some extra features that we would have liked to have included.

Feedback on our requirements and how they were sorted into functional and non-functional was given. Since then, we have reworked the way we thought about said requirements. We had sorted them into more sensible categories and worked to implement them into our application. There were some requirements that we had originally included that we found redundant in the later stages of development, such as the application should not take up more than 15gb of storage space. Since our application was small in terms of storage space, this requirement could have been edited or disregarded to reflect that fact. The team then took these newly updated requirements and began to work on the project with them in mind. Having better thought of and sorted requirements allowed us to code more confidently and acted as guidelines for the programming.

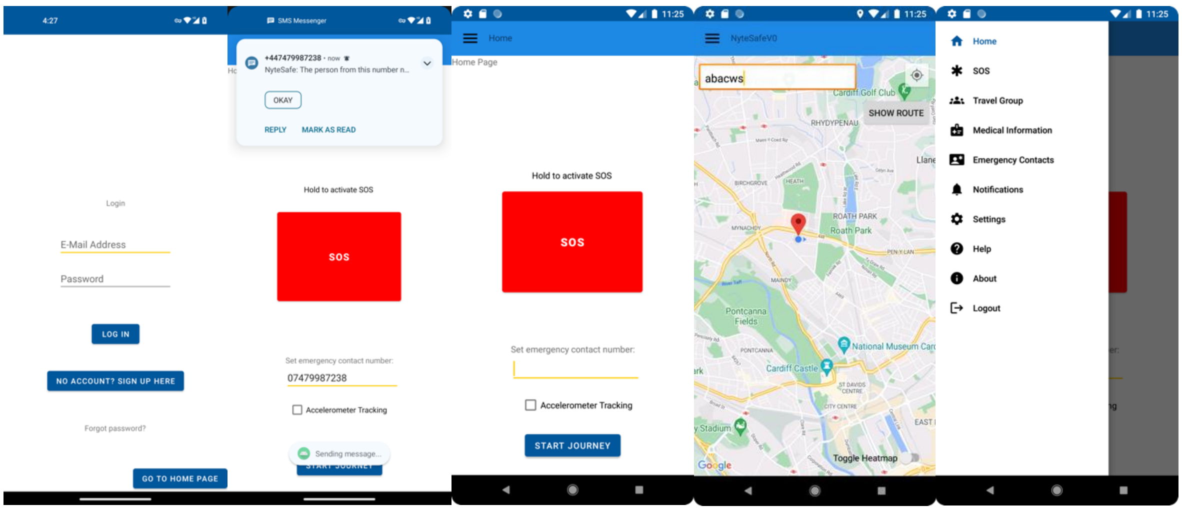
Colour-blindness accessibility was questioned when the client saw that our designs and colour schemes used red and green. It was suggested that we change this to a yellow and blue colour scheme, which is easier to tell the difference between for most people with colour-blindness. Immediately we began to think of designs that would reflect this, and we started to incorporate these colours wherever we could. Doing this would let us avoid having to create a colour-blind setting later down the line, as the application would already be perfectly suitable for someone with colour-blindness.

The client had suggested that we investigate heatmap data for crime statistics. This proved to be invaluable when designing the map feature, as it allowed us to consider areas of high crime and display it on screen. So, to action this we began to figure out how we could integrate this data into our application. Firstly, the team needed to find the data online, and we eventually did. Heatmap data for crime and danger can be found from police websites for cities and towns, and this is where we had obtained the data from. Next, implementing it into the application proved to be a time-consuming task, though it was completed to reflect the client feedback. This will be discussed in more detail later in this report.

Overall, the feedback that we have received from the client has allowed us to create a more functional and accessible application. If we were to carry out this task again, more regular contact with the client would be useful to receive more feedback. In the future, our team would schedule more meetings to display new and updated features that could then be commented on and improved with input from the client. In conclusion, the input and feedback of the client has been extremely useful to aid in the creation and the completion of the project.

Designs

Final Design:


+447479987238 · now
NyteSafe: The person from this number n...

Progress Since Last Report

Our next task after creating the basic design of our app was to try to make it more like the original drafts, we created in the autumn semester. For this project the pages that we were able to fully design and implement are the SOS page, the login page, The navigation bar, and the map page.

Main Focuses

One of our main focuses was to make sure the pages had character and were bright. This is important because apps at this stage, that serve this purpose, normally fail to consider the design elements, making the layout very boring and unremarkable. Only the most successful apps have distinguishable colour schemes and layouts. The main colours we wanted to implement were blue and yellow. The colour blue represents peace and tranquillity, this use of this colour is a common marketing trick to create trust between the consumers. Similarly, the colour yellow is used to represent happiness, and this sparks enthusiasm and interest within the consumer. This draws their eyes and attention to our app. This is important as our users will need to use the app in a time of danger, meanwhile, they are in a vulnerable position for example, being intoxicated. The main difference that we have from the plan is there is less decorative pieces as we did not have time to implement it.

SOS Button

The main function that doesn't fit the main theme is the SOS button. The reason behind that is that we want the user to know where the SOS button is when needed. By having the SOS button nice and large and red, we can be sure that the user is able to press the SOS button in a time of need. Which then lead onto the difference between the draft and the final implementation.

Map and SOS Pages

The difference is that the map isn't on the same page as the SOS page. We discovered that having it on the same page caused the map and the SOS button both functions not able to reach their full potential. The aim for SOS button is to be right in the users face and the map to be able to type in the destination they want to go. By having both on the same page the SOS button isn't in the users face and it is harder to type on the destination onto the map. Therefore, we had to ditch that idea and implement the SOS button as the main function on the main page and the map as a second page. Another reason that we implement the map on a second page is because we assumed that the user wouldn't be too intoxicated when wanting to walk home alone so we implemented the map onto the second page.

SOS Button Continued

Another function that we implemented is the notification when the user pressed the SOS button. The design of the notification is simple and direct. If the user pressed for less than a second, there will be a notification saying it's not activated. Once you press the button long enough it will say message sent. The notification design is simple and direct because we want to clearly tell the user what is happening.

Consistency

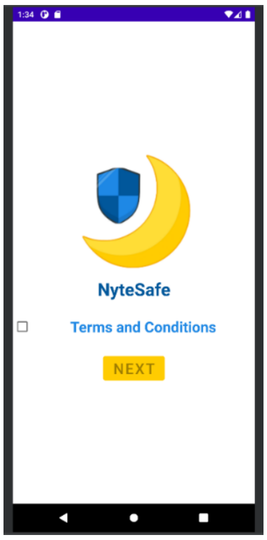
Consistent app design is very important. The main reason is that consistent app design ensures that the app's branding is reinforced every time the user interacts with the app. This is why we tried to place the logo on any page we can and made sure to use the exact same colour codes for the app's design, this builds brand recognition and trust. Also, this makes it easier for the developing process as developers can reuse design elements and templates, leading to increased efficiency.

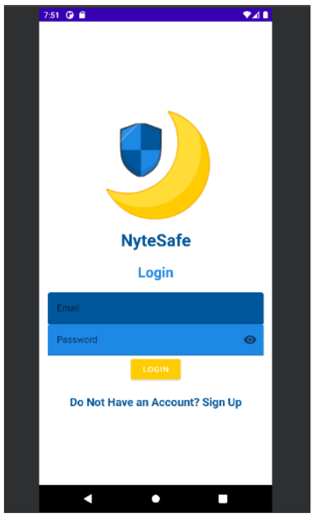
Final Features

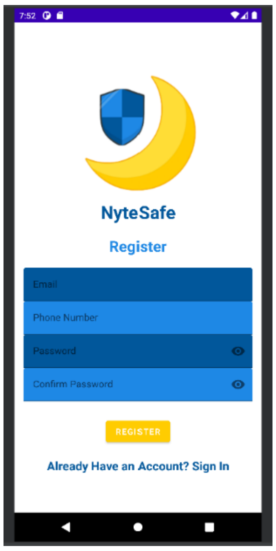
The final pages we created were the terms and conditions, settings page and login/registration in terms of design. The terms and conditions feature a large logo, as this is the first page the user is greeted with upon opening the app. The user is not able to progress to the next page without accepting the terms and conditions. As stated earlier, we tried to implement our colour scheme to enhance the user experience by establishing familiarity. The settings page again features a large logo with the same colours we used previously. One of the options we decided to include was a dark mode, for users to change the lighter background for a darker one. We have also created designs for the login and registration in Android Studio.

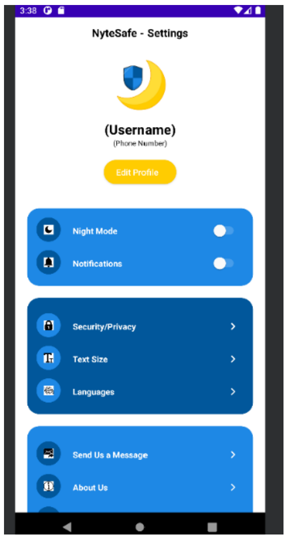
Dark Mode

Dark mode has become increasingly popular in recent years as it has many benefits. For example, it reduces eye strain by reducing the amount of blue light emitted by screens. Also, a dark mode can aid users' readability with visibility issues as it reduces glare and reflections, making text and images stand out more. We are currently working on a toggle to decide between light mode and dark mode. Although, we have tested that the app applies to your native settings. Meaning that if your phone is set to dark mode, the app will be set to this automatically. The images below show the application when the theme is implemented.

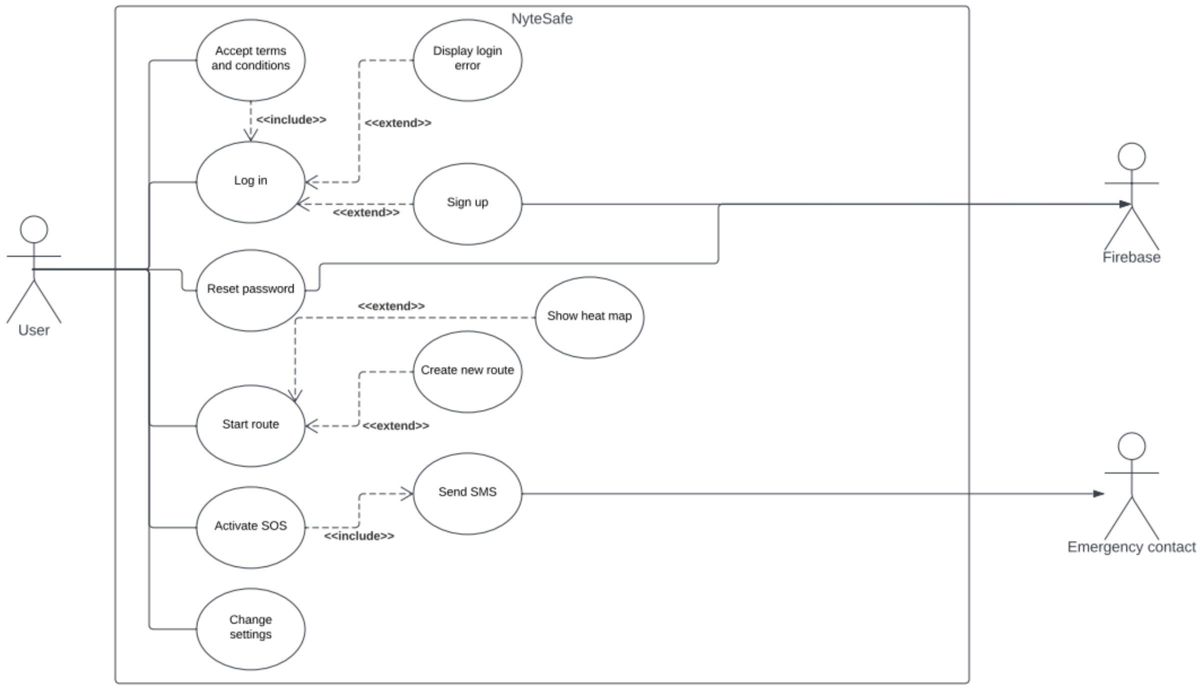








UML Diagram



In our application, the user begins by accepting the terms and conditions. Following this, the user can proceed to log in. The login operation is a crucial part of the application, allowing users to access their personalised settings and routes. It is also designed to extend towards displaying login errors or facilitating user sign-up, the latter of which interacts with Firebase for secure user registration.

In the event of forgotten credentials, the user can choose to reset their password. This feature, like the sign-up operation, communicates with Firebase for secure password management. Users also have the option to alter their application settings via the 'Change Settings' feature. However, due to time constraints, this feature was not fully implemented.

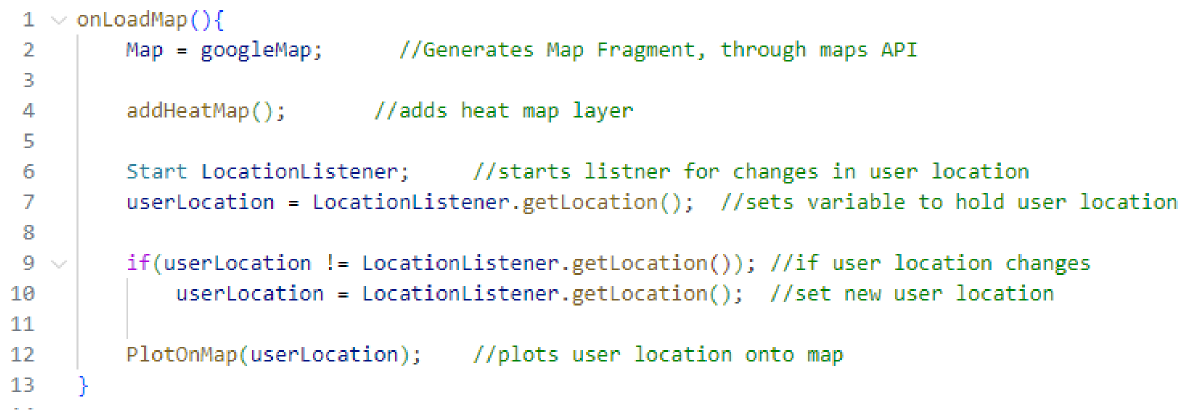
For route navigation, users can initiate the 'Start Route' operation. This operation can be extended to either display a heat map of the route or create a new route based on the user's preference. This provides the user with customizable and dynamic route navigation.

Finally, our application houses a vital 'Activate SOS' feature. In times of emergencies, this feature can be activated by the user to send an SMS to their designated emergency contact, ensuring user safety at all times.

Overall, our UML diagram showcases a user-centric design approach, with a focus on secure user registration and dynamic route navigation, and an emphasis on user safety.

Core Implementation

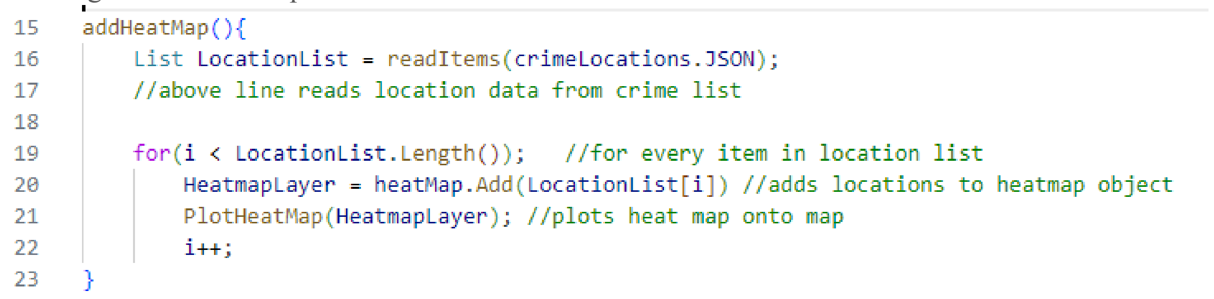
Load Map:



```
onLoadMap(){
    Map = googleMap;      //Generates Map Fragment, through maps API

    addHeatMap();       //adds heat map layer

    Start LocationListener;     //starts listner for changes in user location
    userLocation = LocationListener.getLocation();  //sets variable to hold user location

    if(userLocation != LocationListener.getLocation()); //if user location changes
        userLocation = LocationListener.getLocation();  //set new user location

    PlotOnMap(userLocation);    //plots user location onto map
}
```

A core part of our App is the use of the google maps API. This allows us to display the user's current location on an interactive and dynamic map. We do this by first initialising the map and then requesting the user's location data using a location listener. Our App will also track the user while they are moving and update accordingly.

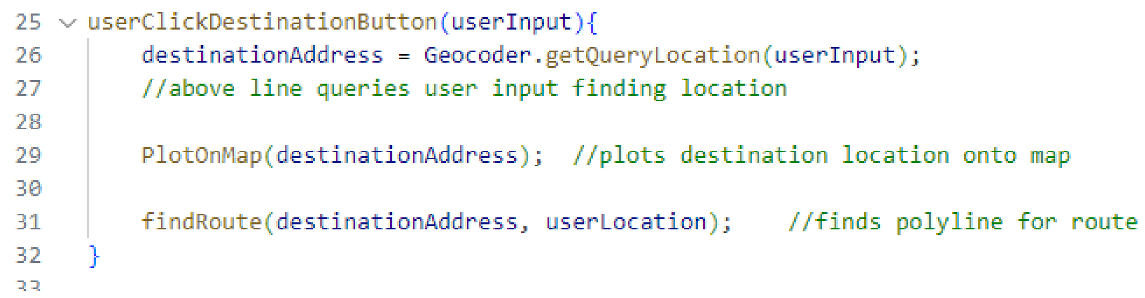
Adding The Heat Maps:



```
addHeatMap(){
    List LocationList = readItems(crimeLocations.JSON);
    //above line reads location data from crime list

    for(i < LocationList.Length());   //for every item in location list
        HeatmapLayer = heatMap.Add(LocationList[i]) //adds locations to heatmap object
        PlotHeatMap(HeatmapLayer); //plots heat map onto map
        i++;
}
```

An additional feature added to our App is a heatmap. layered over the map the heatmap shows the user areas of high crime. This is done by reading crime locations from a JSON file, placing them into a list and plotting over the map.

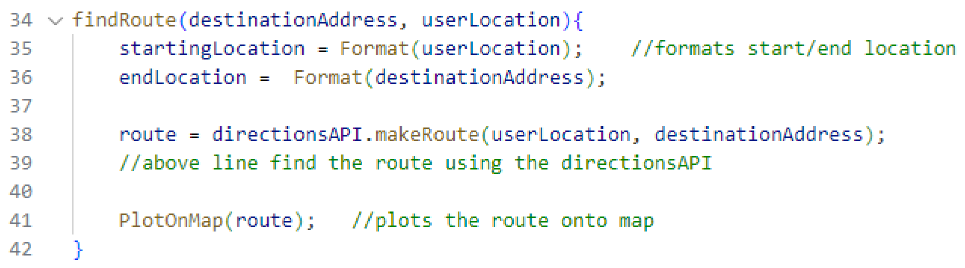
The Destination Button:



```
userClickDestinationButton(userInput){
    destinationAddress = Geocoder.getQueryLocation(userInput);
    //above line queries user input finding location

    PlotOnMap(destinationAddress);  //plots destination location onto map

    findRoute(destinationAddress, userLocation);    //finds polyline for route
}
```

The main focus of our App is to allow the user to route themselves to a wanted location, Here is a rough description of how we do such. Once the user presses the find route button we use the google geocoder to query the user input and receive back the location data for the query, this destination is then plotted on the map and the location data is passed to the findRoute function to generate the polyline.

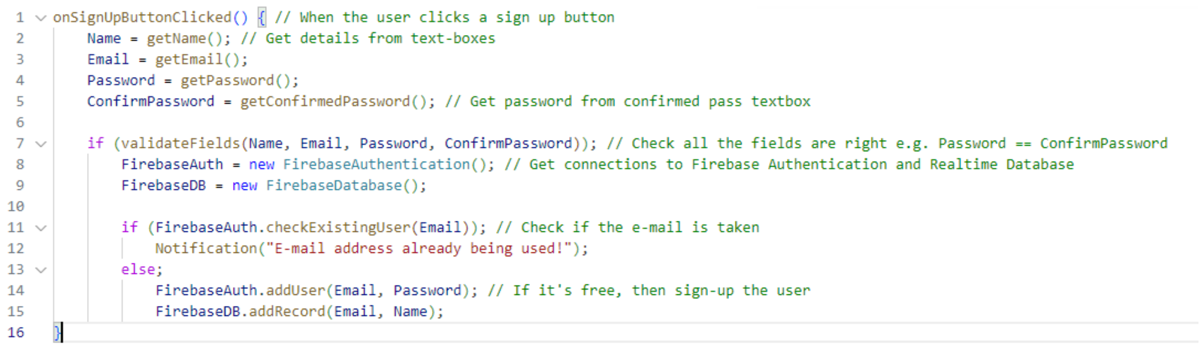
Generating User Routes:



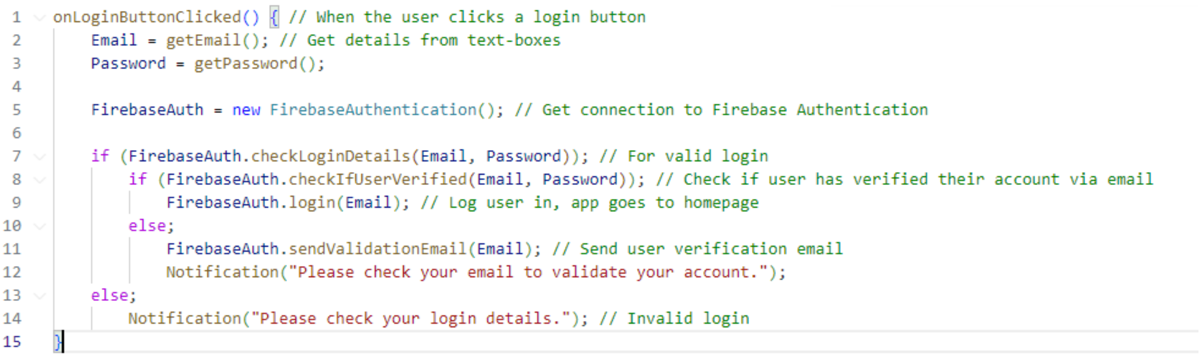
```
findRoute(destinationAddress, userLocation){
    startingLocation = Format(userLocation);    //formats start/end location
    endLocation =  Format(destinationAddress);

    route = directionsAPI.makeRoute(userLocation, destinationAddress);
    //above line find the route using the directionsAPI

    PlotOnMap(route);   //plots the route onto map
}
```

Our app uses the find route function to make polylines that the user can follow to their destination, to do this we use the google directions API. We start by formatting the start and end locations, passing the data to the directions API which will do most of the work. The .makeRoute function will output a polyline that can then be plotted straight onto the map, showing the user the route.

User Account Functions:

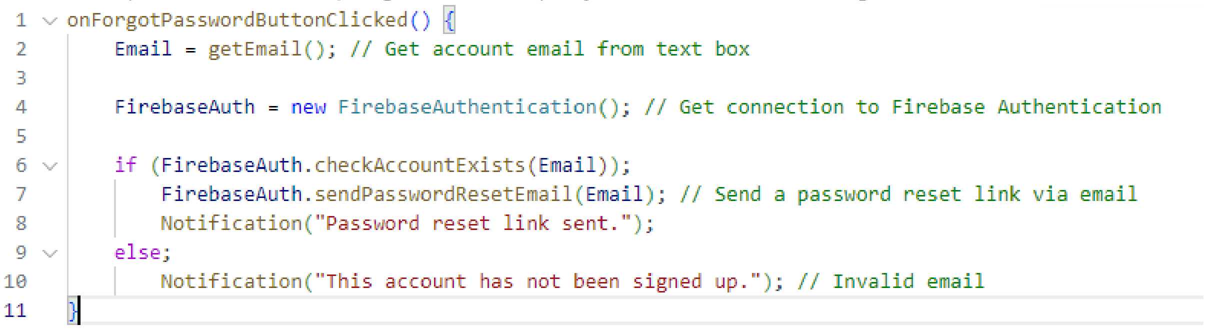


```
onSignUpButtonClicked() { // When the user clicks a sign up button
    Name = getName(); // Get details from text-boxes
    Email = getEmail();
    Password = getPassword();
    ConfirmPassword = getConfirmedPassword(); // Get password from confirmed pass textbox

    if (validateFields(Name, Email, Password, ConfirmPassword)); // Check all the fields are right e.g. Password == ConfirmPassword
        FirebaseAuth = new FirebaseAuthentication(); // Get connections to Firebase Authentication and Realtime Database
        FirebaseDB = new FirebaseDatabase();

        if (FirebaseAuth.checkExistingUser(Email)); // Check if the e-mail is taken
            Notification("E-mail address already being used!");
        else;
            FirebaseAuth.addUser(Email, Password); // If it's free, then sign-up the user
            FirebaseDB.addRecord(Email, Name);
}
```

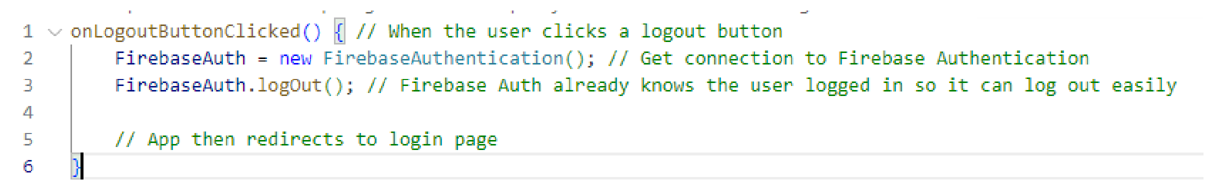


```
onLoginButtonClicked() { // When the user clicks a login button
    Email = getEmail(); // Get details from text-boxes
    Password = getPassword();

    FirebaseAuth = new FirebaseAuthentication(); // Get connection to Firebase Authentication

    if (FirebaseAuth.checkLoginDetails(Email, Password)); // For valid login
        if (FirebaseAuth.checkIfUserVerified(Email, Password)); // Check if user has verified their account via email
            FirebaseAuth.login(Email); // Log user in, app goes to homepage
        else;
            FirebaseAuth.sendValidationEmail(Email); // Send user verification email
            Notification("Please check your email to validate your account.");
    else;
        Notification("Please check your login details."); // Invalid login
}
```

List



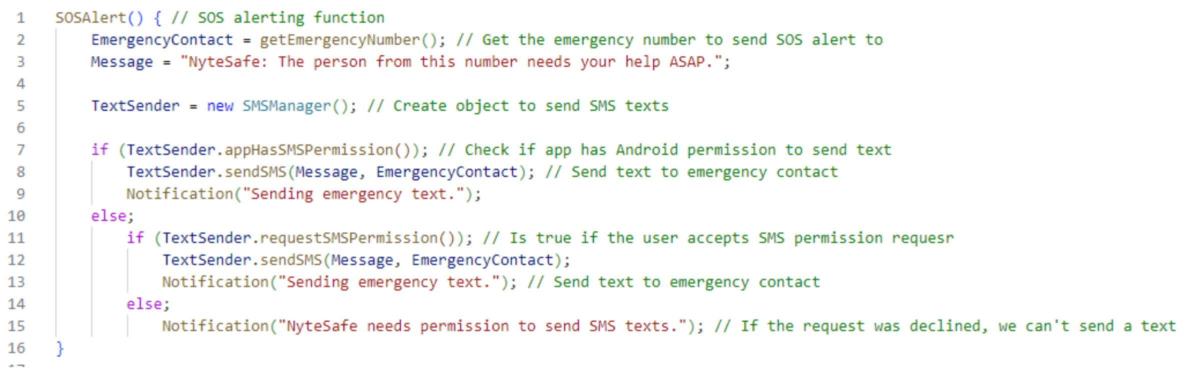
```
onForgotPasswordButtonClicked() {
    Email = getEmail(); // Get account email from text box

    FirebaseAuth = new FirebaseAuthentication(); // Get connection to Firebase Authentication

    if (FirebaseAuth.checkAccountExists(Email));
        FirebaseAuth.sendPasswordResetEmail(Email); // Send a password reset link via email
        Notification("Password reset link sent.");
    else;
        Notification("This account has not been signed up."); // Invalid email
}
```



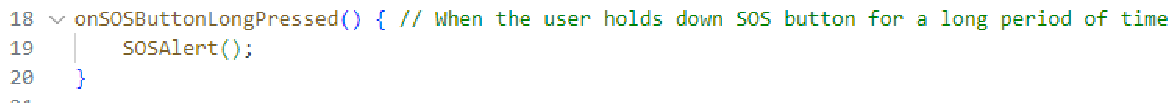
```
onLogoutButtonClicked() { // When the user clicks a logout button
    FirebaseAuth = new FirebaseAuthentication(); // Get connection to Firebase Authentication
    FirebaseAuth.logOut(); // Firebase Auth already knows the user logged in so it can log out easily

    // App then redirects to login page
}
```

The app uses Firebase as the service to store and authenticate user accounts/data. The pseudocode here is a simplified version of how we sign up, log in and send password reset links to users through Firebase. The primary feature is creating instances of different Firebase objects such as Authentication (used for all authentication e.g. signing up, logging in), and Realtime Database (storing other user data e.g. name but not passwords). These objects handle the back-end of account and data handling e.g. hashing passwords, sorting users by automatically generated primary keys, etc. which makes our app account system work well.

SOS Emergency Alert:



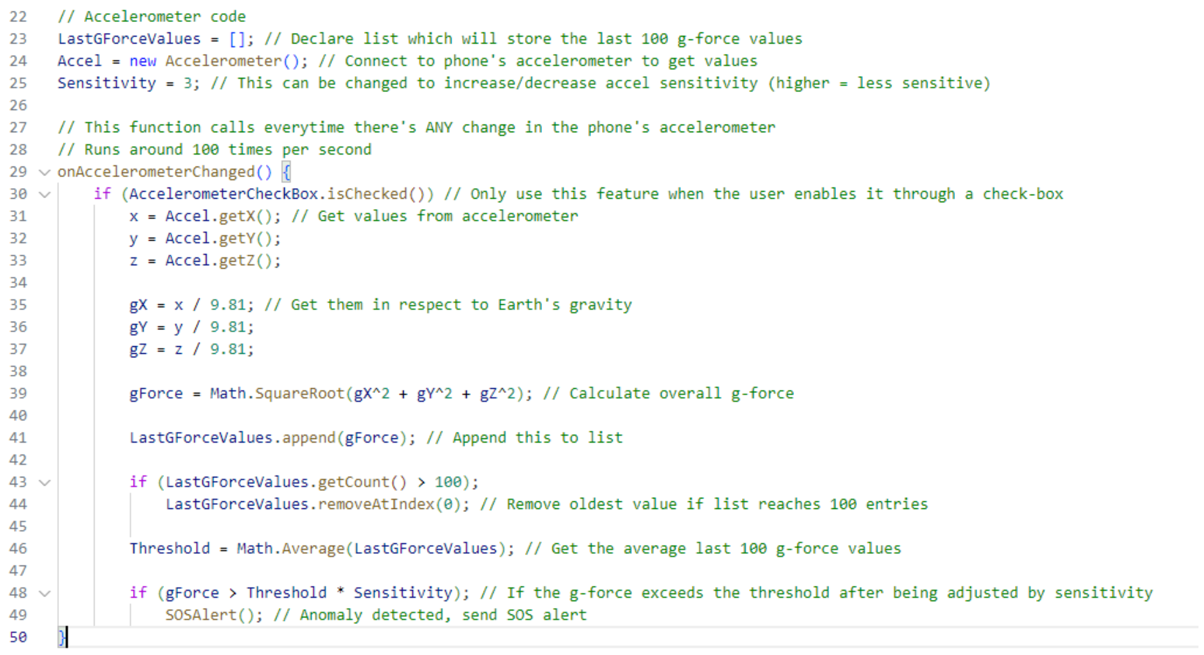
```
SOSAlert() { // SOS alerting function
    EmergencyContact = getEmergencyNumber(); // Get the emergency number to send SOS alert to
    Message = "NyteSafe: The person from this number needs your help ASAP.";

    TextSender = new SMSManager(); // Create object to send SMS texts

    if (TextSender.appHasSMSPermission()); // Check if app has Android permission to send text
        TextSender.sendSMS(Message, EmergencyContact); // Send text to emergency contact
        Notification("Sending emergency text.");
    else;
        if (TextSender.requestSMSPermission()); // Is true if the user accepts SMS permission requesr
            TextSender.sendSMS(Message, EmergencyContact);
            Notification("Sending emergency text."); // Send text to emergency contact
        else;
            Notification("NyteSafe needs permission to send SMS texts."); // If the request was declined, we can't send a text
}
```



```
onSOSButtonLongPressed() { // When the user holds down SOS button for a long period of time
    SOSAlert();
}
```

In the current prototype, when the user long-presses the SOS button, it triggers an SOS alert, which sends a text to the currently set emergency contact number. The app must have SMS permissions beforehand to make this feature work. An extended idea of making this feature more reliable is explained later in the report.

Anomaly Detection:



```
// Accelerometer code
LastGForceValues = []; // Declare list which will store the last 100 g-force values
Accel = new Accelerometer(); // Connect to phone's accelerometer to get values
Sensitivity = 3; // This can be changed to increase/decrease accel sensitivity (higher = less sensitive)

// This function calls everytime there's ANY change in the phone's accelerometer
// Runs around 100 times per second
onAccelerometerChanged() {
    if (AccelerometerCheckBox.isChecked()) // Only use this feature when the user enables it through a check-box
        x = Accel.getX(); // Get values from accelerometer
        y = Accel.getY();
        z = Accel.getZ();

        gX = x / 9.81; // Get them in respect to Earth's gravity
        gY = y / 9.81;
        gZ = z / 9.81;

        gForce = Math.SquareRoot(gX^2 + gY^2 + gZ^2); // Calculate overall g-force

        LastGForceValues.append(gForce); // Append this to list

        if (LastGForceValues.getCount() > 100);
            LastGForceValues.removeAtIndex(0); // Remove oldest value if list reaches 100 entries

        Threshold = Math.Average(LastGForceValues); // Get the average last 100 g-force values

        if (gForce > Threshold * Sensitivity); // If the g-force exceeds the threshold after being adjusted by sensitivity
            SOSAlert(); // Anomaly detected, send SOS alert
}
```

The anomaly detection system uses some maths to get the average g-force of the phone in around the last second. The current g-force is then checked against this average to check for significant changes in the phone's movement, and if there are anomalous changes, it uses the same SOS alert function as before.

Testing

In this section of the report, we will cover 4 main points; our approach to testing, how we tested from both a developer's and user's perspective, the test reports we created, and finally what the final product does well and where it starts to fail.

Approach

The approach that we took to testing was a multi-phase approach. We chose to have two main phases, these being in-house testing and then open beta testing. The in-house phase of testing was completed by the development team and was in line with our AGILE approach to the project. This allowed us to thoroughly test each new feature as it was implemented, meaning we constantly kept the testing workload as low as possible as we were testing as we developed. This testing was all recorded and tracked using a test case form, you will find this document attached as an Appendix. These test cases allowed us to track the bugs found, as well as the steps that led up to the discovery of the bug. This allowed us to recreate the bug in order to validate it was an issue and then test we had fixed it. The form also allowed us to easily track any specific features that are causing errors as we were able to find patterns in the reports once they were all collated into one document.

Our second phase approach to testing was to use open beta testing with users being given specific instructions on how they can test the app. You will find this letter attached as an Appendix. The purpose of the open testing was to allow us to collect the data recorded during the testing and usage of the app so we can analyse it for anomalous behaviour, which we can then use to refine the parameters we use to classify route deviation, and dangerous device movement. The beta testers also had access to a simplified version of the test case form so that they can submit any bugs they encounter to the development team with as detailed an explanation and images as they can manage, which allows us to recreate the bug and troubleshoot it.

User and Developer Perspective

In order to ensure our testing was as thorough as possible we made sure to test the app from both a user's and a developer's perspective. This was mainly done through our use of open beta testing, as well as asking friends, family, and housemates to test features of the application, as well as the application as a whole. We mainly tried to focus the user testing on the actual usability and feel of the app, for example asking people to feedback on the design of the application aesthetically as well as the way the app was laid out. This allowed us to make changes to make the app to make it more user friendly and accessible. Whilst bug feedback was useful from a user perspective the lack of technical knowledge from the majority of the testers meant their feedback on the design was much more useful. When it came to developer testing, we were almost entirely focused on trying to break the app, stress testing and using it in a way we didn't plan originally. This was so we could find bugs and errors with the code in order to fix them. Having an understanding of how the app works, on a software level, meant we were in a much better position to find flaws in the app and then subsequently fix them.

Test Reports

To keep track of how we were progressing with testing, each time we found a bug we would log it and create a test case using the bug form we created. This form asks for as much detail as possible to allow recreation of the error. It first asks for a summary of the scenario that the bug occurs in, for example creating a new user account. It then asks for the steps that were taken leading up to the bug, which again allows us to recreate the bug. Next it asks what prerequisites are required to cause the bug, as well as the data that will be required to recreate it, for example it may be a specific email or location. Finally, it asks what the expected outcome would be should the bug have not occurred, to stick with our example the expected outcome would be the creation of a new user account. Using these forms then allows a developer to recreate the bug, fix it, then run the scenario again and note down the actual result and whether it passed or failed the test. Each time it fails if anything has changed in the process, a new test case is created, and the process is repeated.

The Good and The Bad

In this final section we will discuss the areas that our app does well in, as well as covering some points that it does not. The first and most relied upon point is the user login/account system and supporting backend software, without this we wouldn't be able to have any users or be able to do any calculations using their location data. This system works so well thanks to the firebase facilities we chose to use which handle user creation, verification, and password resets for us. The next part that our app does well is the crime heatmap, and subsequent avoidance of these areas. This helps towards solving the main section of the problem the client presented to us. This system works using our route planning system and the heatmap itself, which maps all the relevant crime data around the user in a manner that allows them to clearly see areas that should be avoided. Finally, our app handles anomaly detection very well during the user's journey, this includes deviation from the route provided as well as any potential altercations or injuries that may occur to the user. The route deviation works by calculating how far a user travels off the given route, and for how long before it alerts the user and their emergency contact. The altercation detection works by using the device's accelerometer to detect readings that we classify as anomalous. We worked out what data should be seen as problematic by performing numerous tests and logging the data outputted by a device's accelerometer. One area that the app begins to fail is with the route planning avoiding areas with a high crime rate, sometimes the route will avoid these areas and take a user along the assumed safest route. However, in some edge cases it will take the user through the areas with past criminal activity. Due to the time limitation, we were unable to find the route cause for this issue and resolve it, however it would have been our pain priority if we had more time.

Justification/Evaluation

Overview

Overall, we feel as though we have designed the app in the most logical and efficient manner. By dividing tasks across all nine group members and through strong collaborative efforts, we managed to create an app which services as many requirements as possible and functions properly and consistently. The user interface of the application was at the forefront of our minds when designing this project. We decided a simplistic and clear approach would be most effective, in order to be usable by our target audience.

Design Methodology

During the planning phase of the project, we decided to use an AGILE methodology. This meant we had features for each version of the app to focus on, with each iteration of the app. We successfully completed all the features of the v0 and most of v0.1. We could not unfortunately implement a grouping users feature and the key location. Initially we tried to implement a Bluetooth connection between users within the same group. This approach was soon scrapped as the Bluetooth connection would have a range too short and the process of connecting to other devices safely had too many issues. The key locations feature which would allow the user to save locations on the map which could quickly set as the destination for the route pathing. We did not implement this due to time constraints however it would be quite easy to.

Integration Issues

Throughout the development of the application, we came across many issues with integration. Android Studio contains a build automation tool called Gradle which manages project dependencies, compiles source code and package applications. When the projects are uploaded to the git repository with the build folder, variables will be changed once cloned from the same repository. This meant when new updates to the main code were made and team members needed to update their own version, they'd need to fix Gradle each member with their own different errors. This slowed down development and deterred us from sharing our versions of the files. Instead of using git version control, opted to send each other the files as zips. This meant the variables would not change.

Colour Scheme

The final application does not contain a colour scheme and theme that matches the concept designs that we prototyped earlier in development. This was due to time constraints and the focus of development shifting toward the functionality rather than the aesthetics in order to have a functioning prototype in our time limit.

Notification

We decided on having an SMS being sent to the emergency contact once the anomaly detection is set off. We changed this from an email or phone call because we thought that sending off a text message would be more discrete and therefore safer for the user in question. The anomaly detection system was initially supposed to be based on a machine learning model however we have opted to use averages and test data from measurements of a real-life simulation. We decided basing the test data on an extreme example of an anomaly being detected to ensure that setting off the feature accidentally would be unlikely.

Route Planning

The route planning system works effectively and has a fully functional crime heatmap overlay. We wanted the heatmap to be adjustable to the screen size of the map to give the user the option to zoom into areas and give a more accurate representation of the streets and neighbourhoods. This was implemented over the original map code with a toggle button. The toggle feature meant the user could turn it off and better see the map with the route when needed.

Overall Strengths

Overall, I feel as though we have a project which flourishes in a lot of areas. For one, we have a fully capable login system, which allows a user to reset their password at will. Whilst this was a very early part of our system, our ability to ensure that it was created to be efficient and functional, as well as secure, is a very strong point of our project. Second, the full integration of the google maps API into our project is an increasingly strong part of our project. At the start, we had many troubles with both securing and amalgamating it into the application. However, with time and effort we managed to negate these issues, leading to us having the program utilise google maps for route planning. This means that our map is reliable, constantly updated and stretches worldwide. Finally, another one of our strong points we would argue is our implementation of crime data into the project. Through research, we managed to find data on crime in Cardiff. Through use of python code, we managed to filter this data and implement a heat map onto our app, which shows the user high crime areas and where is safer for them to go. As a result, the app is effective in increasing the safety of the user, satisfying its main purpose.

Limitations

Our app does have some limitations however. For one, although the user can see where crime is, we were unable to implement the route divergence accordingly. This means that it is up to user discretion to use the map to avoid the high crime areas, as routes generated can bring the user directly through high crime areas. Obviously, this is a limitation, as it means that user safety isn't fully maximised. Second, we require reception for our app to function. Within some areas, especially in Cardiff, reception for phones is low, meaning that the app isn't able to function as it should. Whilst we planned for offline compatibility, we were unable to get to a stage where we could implement it. The user therefore cannot access the app at all times, which in turn means that their safety is dependent on reception, a flaw in our app.

Meeting Client Requirements

We would like to think that the app meets the clients' requirements moderately well. The primary aim of our project brief is to make 'travel at night safer', which our app does - it gives the user a route home and alerts them to areas which have higher crime rates. We also correctly created code which transmits the coordinates of the user to the central server, again following the directions of the project brief. At a fundamental level, we absolutely implemented the basic tasks and functions imposed on us. However, we did omit some parts of the brief. For one, we were unable to implement the idea that if a user deviates from the route too much, or doesn't arrive on time, the police would be alerted. Mainly, this was due to two reasons. First, we were unsure how to implement such a function to where there wouldn't be nuisance calls to the police. People may walk slowly, stop for a kebab, or stop at a friend's house, meaning that they would still be safe, even though the police would be alerted. Second, we simply fell to time constraints. In future, these would be things we'd look to implement, in order to fully adhere to the project brief. Thus, we feel that in this current build, we have moderately satisfied the client's needs.

Legal, Social and Ethical Issues

We were very influenced by legal, social, and ethical issues within our project. First, legal. In order to adhere to all legal issues, we ensured that the user had to agree to terms and conditions prior to usage of the app, due to the fact that it collects personal location data. We also ensured data is stored in adherence with the GDPR and made sure that users were able to delete accounts, along with all associated data with said account. Social issues we negated through the use of terms and conditions also, as we ensured we had user consent to collect data. Further, when applying for ethical clearance from the university, we maintained a fair social policy by asking users to test from places which weren't sensitive to them, such as their homes or places of work. Last, ethical issues. Through lengthy correspondence with the ethics board, we managed to achieve ethical clearance for the project in early May. To fully adhere to ethical issues, we created consent forms which were made to be incredibly clear, so that people testing the app would be fully aware of all data taken. They also could ask to stop at any time, meaning they weren't pressured at all. Overall, legal, social and ethical issues shaped our project in that they ensured the protection of the user's rights and data was done so in a high-quality manner.

Future Work Required

Ability to add stops enroute

Our group could have included a feature for adding multiple stops during a journey by updating the user interface. We would add a '+' button for users to input additional stops. Then, we would adjust the backend to consider these stops when planning the route using the Google Maps API. We would ensure that the updated route accounts for these additional stops and the navigation instructions are adjusted accordingly.

Show estimated journey time

To display the estimated journey time, our group would modify the app to request this information from the Google Maps API. We would then calculate the total journey time by adding up the durations of each route segment. This total time would be displayed on the app interface, either close to the start and end points or within the navigation instructions.

Buttons need to find the Goldilocks zone on size

Our group would test various button sizes and layouts to find the optimal size for different user scenarios. We would gather feedback from users about the ease of use and analyse the data to identify preferences. Using responsive design principles, we would create adaptable interfaces that work well with different screen sizes, resolutions, and orientations.

Dark mode and Light mode

To implement dark and light mode, our group would design alternative colour schemes for both modes. We would ensure that text and background elements have sufficient contrast for readability. We would then add an option in the app's preferences for users to switch between modes, either manually or automatically based on the device's ambient light sensor. We would thoroughly test the app to ensure a smooth transition between modes and the visibility of all UI elements.

Works without signal

To make the app functional without a signal, our group would implement offline map caching using the Google Maps API. We would store map tiles for offline use and adjust the map rendering logic to use these cached tiles when the network connection is unavailable. We would also modify the app's data handling logic to store user data locally when offline and synchronise with the Firebase database when the connection is restored.

Works without internet connection

Along with offline map caching, our group would integrate other offline features such as local storage for user data, offline route planning, and emergency contact functionality. For offline route planning, we could use a third-party library that performs routing calculations on the device. We would store user data and emergency contact information locally on the device. Our team would design the app's logic to handle changes in network connectivity and gracefully transition between online and offline modes. We would then test the app under various connectivity scenarios to ensure a seamless user experience.

An Attractive UI

In future, we would look to also implement a more attractive UI design into our application. We had developed some prospective designs, which utilised more of our blue and yellow colour scheme that we had hoped to integrate into the application prior to the end of the project. However, with time constraints and the issues we faced with android studio, we were unable to reach a stage where implementation of this would be a possibility. Obviously with more time, we would reach said stage and implement accordingly.

iOS Compatibility

Another thing we would potentially look into in future would be iOS compatibility. Based on the fact that android studio seemed to be simpler to code with, we opted to use this and create a solely android program. However, were we to get into a position where our android app was fully functioning and working as intended, we would certainly engage with the potential prospect of creating an iOS functional app.

Google Wallet Integration

Another feature we ended up cutting from our original plans but would ideally implement if we had more time is a feature which would flag suspicious use of the users' Google Wallet, which is basically what android uses to allow a user to use their bank cards from their phone. This was an idea raised to us by the client, however we never really found a suitable method of measuring the payment usage. As a result, we couldn't find a suitable way to correctly include this in our project. Although, had we more time, we would likely search further for a method, and place it in the program.

Medical Information Page

One of the other ideas we had which we were tempted to include was a medical information page. Smartphones give the option for the users to store their emergency medical data, so that it can be used in an emergency. Given the context of our app, we felt as though integrating this information into our project would be a smart and suitable thing to do. However, time was against us, and we decided to cut this based on the fact that other tasks held a higher priority. In future, this would be one of the features which we would prioritise, on the basis it would be simple to implement and highly beneficial to the user.

Hardware Button SOS Activation

Lastly, one feature we would look to implement would be one which would allow the user to send an SOS message when pressing their power button three times in succession. This would allow them to summon help when in a perilous situation, such as an event when being mugged for instance. By pressing the power button 3 times, the user would trigger the SOS function in the project, which would alert local authorities to the fact that they were in need. This would have the same functionality as the SOS button in the app, but having the three presses act as a trigger would allow the user to have multiple ways of activating it. Also, the choice for three button presses was done to minimise the chance of accidental activation, meaning the authorities wouldn't receive false alerts regarding the safety of the user.

Conclusion and Closing Thoughts

To close the report, we will discuss some of our thoughts and conclusions from completing the project, and what we have learned.

Android Studio Learning Curve

When we started the project by far the largest issue that we encountered weas using android studio, at the time java as a language was still very new to us, and Kotlin, the language used in android studio, is not an exact copy of Java so we needed to learn how tom use the language. We also had to get used to using Gradle and the issue it presented as discussed within the report.

Large Group Work

We also had to get used to working as a large group on a difficult ad long project. It took as time to get adapt to splitting the workload evenly and playing to the strengths of everyone within the group. However, once we did get into a stride as a group, we were able to very quickly make progress very quickly and work well as a group.

Mobile App Development

The final issue we encountered was working on mobile app development, we wrongly assumed it would be similar to web development, however this proved to not be the case. We had to work with new file types we had never seen before such as XML files and activities. This proved to be a valuable experience and give us a very useful skill set, however it presented a lot of issues at the start of the project.

Appendix

Appendix i – Test Cases

Appendix ii – Letter to Beta Testers