Interim Report

Group 07 (CM2305 Group Project)

Personal security system for night-time economy
Project Title: PAL—Peace And Love

Team Members:

[1739635] Aaron Smith
[1800857] Henrijs Princis
[1826905] Euan Morgan
[1827049] Benjamin Eddy
[1834661] Louis Davies-Cren
[1838838] Marley Sudbury
[1880917] Sara Abidi
[1969507] Lizheng Huang

Client:

Kiril Sodorov (SidorovK@cardiff.ac.uk)

Supervisor:

Mohd Syafiq Bin Zolkeply (ZolkeplyM@cardiff.ac.uk)

Table of contents:

Introduction

Our client approached us with a brief that entailed developing a personal security system to help reduce night crime in Cardiff. The brief specified that we should allow users to set a start and end location, and their GPS coordinates should be used to monitor the users' progress. In our initial meeting with the client, we agreed on changes to expected features that were originally specified in the brief; this is due to certain concerns including the practicality and ease-of-implementation of the features.

This problem is particularly interesting due to the sheer volume of sexual offences and violent crimes that happen in the UK, especially in Cardiff. According to a national survey of Crime in England & Wales, "20% of women and 4% of men have experienced some type of sexual assault since the age of 16" (Office for National Statistics 2017). Focusing on Cardiff, we see that violence and sexual offences makes up the largest proportion of reported crimes (25.37%) between November 2018 and October 2019 (South Wales Police 2019).

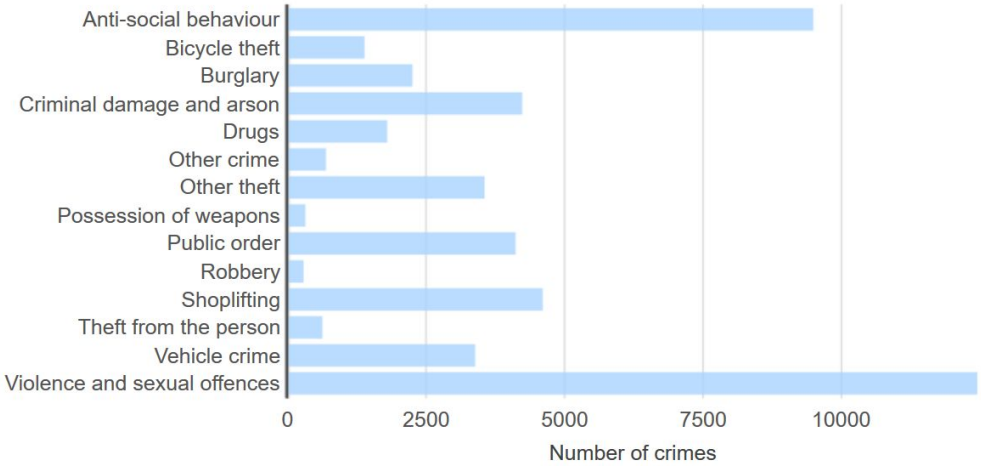


Figure 1—Statistics on the number of crimes, based on the type of crime, between November 2018 and October 2019
Source: South Wales Police (2019)

In this report, we will discuss the decisions that have been made, and elaborate on the progress since the beginning of the project. The majority of the completed tasks currently consists of communicating with the client and decision making. Although we are yet to complete a large proportion of the development, we have already achieved a certain amount of rapid prototyping. We decided this was necessary to be able to progress (as

seen in the evidence). Throughout the project, we have worked collaboratively to find a suitable solution that can be implemented within the time made available to us.

System Requirements

The system requirements define core aspects of the system. They are broken down into 2 sections: functional and non-functional requirements. Functional requirements are what the customer wants the system to do, clarified by testable acceptance criteria. Non-functional requirements define aspects that the user needs the application to perform to enable the functionality to operate effectively.

Process of identifying system requirements

The team had a dedicated sub-team of 4 members to identify and explain the system requirements to the rest of the team. The sub-team had communicated in team meetings to identify core aspects of the system and categorise them into must, should, could and won't (MoSCoW format). Next the sub-team had split off to complete their allocated section. Once completed, the sub-team presented their ideas to the team, and agreed on a final version of the system requirements.

Functional Requirements

Must - Functionality the system must contain upon completion to allow the application to fulfill its purpose

1. The user is able to create an account

Acceptance criteria:

Demonstration of:

a. The user is presented with a login page within 5 seconds of launching the application.
b. On this page there is a 'Create an account' button.
c. Upon the user selecting the button, they will be prompted for their name, phone number, age, and password.

2. The user is able to log in to the app

Acceptance criteria:

Demonstration of:

a. The user is presented with a login page within 5 seconds of launching the application.
b. The user will be prompted for their phone number and password.
    i. If login details are correct, they are granted access.

ii. If login details are incorrect, they are shown an error message and prompted again.

c. Upon the user selecting the button, they will be prompted for their name, phone number, age and password.

3. The user must add at least 2 trusted contacts before being able to use the app

Acceptance criteria:

Demonstration of:

a. The user is prompted for a minimum of 2 and a maximum of 5 trusted contacts when first logging in to the application.
b. The trusted contacts will receive a text message within 20 seconds of being added, informing them that they are a trusted contact.

4. The user must be able to edit their account information.

Acceptance criteria:

Demonstration of:

a. On the homepage there will be a 'My Account' button.
b. Upon clicking this button, the user will be presented with their information within 3 seconds.
c. The user can then select 'Edit' next to the current field they wish to change.
d. The system will store new details accurately 99% of the time.

5. The user can input a destination and receive an appropriate selection of routes.

Acceptance criteria:

Demonstration of:

a. The user can input a street address/ postcode as a destination.
b. If the input is valid, they are shown a set of potential routes.
c. If the input is invalid, they receive an error message and are prompted for the address again.

6. The user can select the route they would prefer to travel, based on time vs safety.

Acceptance criteria:

Demonstration of:

a. Upon entering a destination, the user is presented with one or more safe routes within 10 seconds.
    i. Providing they have a consistent internet connection with a download speed of 1MB/s+
b. The user can then select their preferred route and the app will begin tracking them within 5 seconds, accurate to a 0.1 mile radius.

7. The user is able to pause the journey if they need to stop.

Acceptance criteria:

Demonstration of:

a. While on an active route, the user can select the 'Pause' button to pause a journey.

b. The user will then be prompted for a passcode or biometric data to verify their identity.

8. The user is able to cancel the journey.

Acceptance criteria:

Demonstration of:

a. While on an active route, the user can select the 'Cancel' button to cancel a journey.
b. The user will then be prompted for a passcode or biometric data to verify their identity.

9. The police are alerted if the user substantially diverges from the previously chosen route.

Acceptance criteria:

Demonstration of:

a. The police are alerted if the user diverges from their path by 25 metres and fails to cancel the 'Contact Police' functionality.

10. The user is able to stop the 'Contact Police' functionality.

Acceptance criteria:

Demonstration of:

a. Upon detecting a 25-metre divergence the app will notify the user and begin a countdown.
b. The user can enter their passcode within the countdown limit to stop the applications functionality from notifying the police.

11. The user is able to modify the timer for contacting the police.

Acceptance criteria:

Demonstration of:

a. Within the 'Settings' page of the application, the user can adjust the timer for how long the countdown is until the police are notified.

12. The user can send GPS coordinates to the central server.

Acceptance criteria:

Demonstration of:

a. If the user presses the 'Panic' button, their current GPS coordinates are sent to, and stored by the server (accurate to a 0.1 mile radius).

Should – Functionality the system should contain to allow a pleasant user experience.

13. The user can press a dead man's switch when they are worried about their safety.

Acceptance criteria:

Demonstration of:

a. Whilst on an active route, there will be a “Dead Man’s Switch” button visible on screen.
b. When the user removes their finger, the (cancellable) countdown to contact the police is started.

14. The user can change the colour scheme.

Acceptance criteria:

Demonstration of:

a. In the settings, the user can select a ‘Colour-Blind’ colour scheme to accommodate any visual disabilities

15. The user can access a ‘Help’ page.

Acceptance criteria:

Demonstration of:

a. A help section is available in the main menu of the app.
b. When selected, the user is directed to a page within 3 seconds, which contains a button for each section.
c. When the user selects the button with the label they wish to receive help on, they are directed to another page within 3 seconds explaining usage of that functionality.

16. The user can access a ‘Contact Us’ page.

Acceptance criteria:

Demonstration of:

a. A ‘Contact Us’ section is available in the main menu of the app.
b. When selected, the user is directed to a page within 3 seconds, which provides a contact number, office opening hours, and an email for the app’s development team.

17. The user can see a colour-coded heat map of crime data along a route.

Acceptance criteria:

Demonstration of:

a. A colour coded heat-map of crime data for a route should be overlaid when a journey is active.
b. This data should be automatically updated on the first day of every month, for crime data on each road from the last 3 months.

18. If the user does not arrive at the destination at the estimated time, the police are informed.

Acceptance criteria:

Demonstration of:

a. If the user has not reached their destination within 20 minutes of the expected arrival time, the cancellable ‘Contact Police’ functionality will begin.

19. The user can change the destination of their journey after starting.

Acceptance criteria:

Demonstration of:

a. When on a route, the user should be able to change their destination without terminating a journey.
b. After inputting a new destination and clicking the appropriate destination address, the user will have to input their passcode or biometric data to confirm their identity.
c. The destination will then be updated to the new location.

20. The user can be put into a group of people taking a similar route.

Acceptance criteria:

Demonstration of:

a. If the system detects 3 or more users travelling on a high crime (rated yellow/ red) road, the users receive a notification stating that they can join a group.
b. If the user accepts, the app suggests a meetup point.
c. If they decline, they will continue the journey individually.

Could – Functionality the system could include if our time limit allows, to allow convenience for the user.

21. The user can allow the app to capture and store video and audio footage when a threat is perceived.

Acceptance criteria:

Demonstration of:

a. The user can enable/disable this function in the settings.
b. If enabled, and the app detects a threat via gyroscope or the "dead man's Switch", it will capture video and audio from the microphone and camera until the user inputs their passcode.

22. The user is warned about low battery.

Acceptance criteria:

Demonstration of:

a. When the app is in use, the user will be notified when their battery reaches 20%.

23. If the user is on a route and their phone detects atypical movement the 'Contact Police' functionality is triggered.

Acceptance criteria:

Demonstration of:

a. If the phone detects shaking or spinning that is irregular, the countdown timer to phone the police will begin.

24. The user gets a notification when a dangerous road is in close proximity.

Acceptance criteria:
Demonstration of:

a. When the user is within a 0.1 mile radius where crimes have been reported on more than 54 of the last 90 days, they will receive a notification warning them it is a potentially dangerous area.

25. The user can see roads with popular nightclubs/bars where the crime data may be inflated due to antisocial behaviour within the bar.

Acceptance criteria:
Demonstration of:

a. When on a route, the user can see locations of popular nightclubs/ bars for human judgement on crime data.

Won't – Functionality the system won't include due to time constraints.

26. Users have a rating which determines how trustworthy they are.

Acceptance criteria:
Demonstration of:

a. The app's external database will load the users ratings (rated from 1 to 5).

27. The user can anonymously rate users they are grouped with.

Acceptance criteria:
Demonstration of:

a. An option to rate users that a user is currently grouped and on a route with is presented in the 'Peers' section.
b. The user is prompted for a score of 1 to 5 for the user.
c. This score is then stored in an external database.
d. A user cannot submit a rating for another user more than once.

28. The user can see the ratings of users they are about to share a route with.

Acceptance criteria:
Demonstration of:

a. An option to view ratings of other users is available in the 'Peers' section.
b. When the user is asked to join a group, they should see this 'Peers' section on the navigation page.
c. The user's review data should be updated every 30 minutes.

29. The user will be banned from the app if they have a low rating.

Acceptance criteria:
Demonstration of:

a. If a user's average rating after 10 reviews is lower than three, they will receive a message saying "You have been banned from the app due to low ratings."
b. They will then be unable to use the app's functionality.

   c. The app should ban the user within 10 minutes of their rating dropping below the threshold.

Non-Functional Requirements

Non-functional requirements are what the user needs the system to have the ability to do for a usable system. They are often testable, quantifiable, and can relate to the system's security and maintainability.

1. The system's database must be secure from all third parties as sensitive GPS data and personal data is being stored.
2. The system should load the location route within 10 seconds of the user's request, on a compatible device containing GPS.
   a. Providing the user has a consistent internet connection with a download speed of 1MB/s+.
3. The app should be maintainable; ability to change code to meet the client's requests by having access to a version control service that stores the code.
   a. The source code should be backed up on one physical device and a cloud service/version control system to prevent losing any changes.
4. The system should update its crime data by pulling it from South Wales Police website on the first day of every month to ensure relevance.
5. The interface should be simple and easy to navigate:
   a. Easy to read font.
   b. Clearly labelled buttons.
   c. All menus must have a consistent layout.
      i. E.G. Common buttons maintain their colour and positioning within the different application page's.
   d. 90% of users should understand how to use the app within 5 minutes of their first use.
      i. The app will contain a help section to provide the user with detailed usage information on the application's complete functionality.
6. All accounts created must only be able to submit strong passwords.
   a. 8 or more characters.
   b. Mixture of upper/lower case letters, numbers, and symbols is required to make a password valid.
7. The application should run smoothly on all mobile devices.
   a. The applications start-up cycle should last no more than 5 seconds on a compatible device with GPS.
8. User data must be stored securely.
   a. Passwords must be encrypted.
   b. All user data should be backed up weekly.
9. All user input must be parsed correctly to ensure there are no unhandled exceptions for incorrect input.
10. Accessibility:
    a. The app should accommodate for people with specific vision disabilities.
       i. An option for different colour schemes for colour-blind people.

   ii. An Option to display the UI in a larger font.
   iii. Possible text-to-speech to read aloud any text displayed (if time constraints allow).
11. The app should have 99.9% uptime, excluding cases where the map API being used for the application is down.
12. The app should detect potential users to group with 90% accuracy and efficiency.
13. The app should not use more than 10% of the phone battery every 30 minutes.
    a. The app could warn the user to use battery saving mode to minimise the battery consumption. This will include:
       i. Lowering screen brightness.
       ii. Turning off Wi-Fi.
       iii. Turning off Bluetooth.
       iv. Lowering screen resolution
14. The app should take up less than 200MB of storage space on the device.
15. The app should not crash on more than 2% of actions within a one-hour time limit.
16. All users of the application must be 18+.
    a. Due to grouping functionality, the users must verify their age by entering their date of birth when creating an account.
17. All sensitive data such as passwords must be obscured when the user enters the information into the application.
18. All of the applications interfaces (excluding the 'Start a Journey' functionality) must have a maximum loading time of no longer than 3 seconds.

Use case diagram:

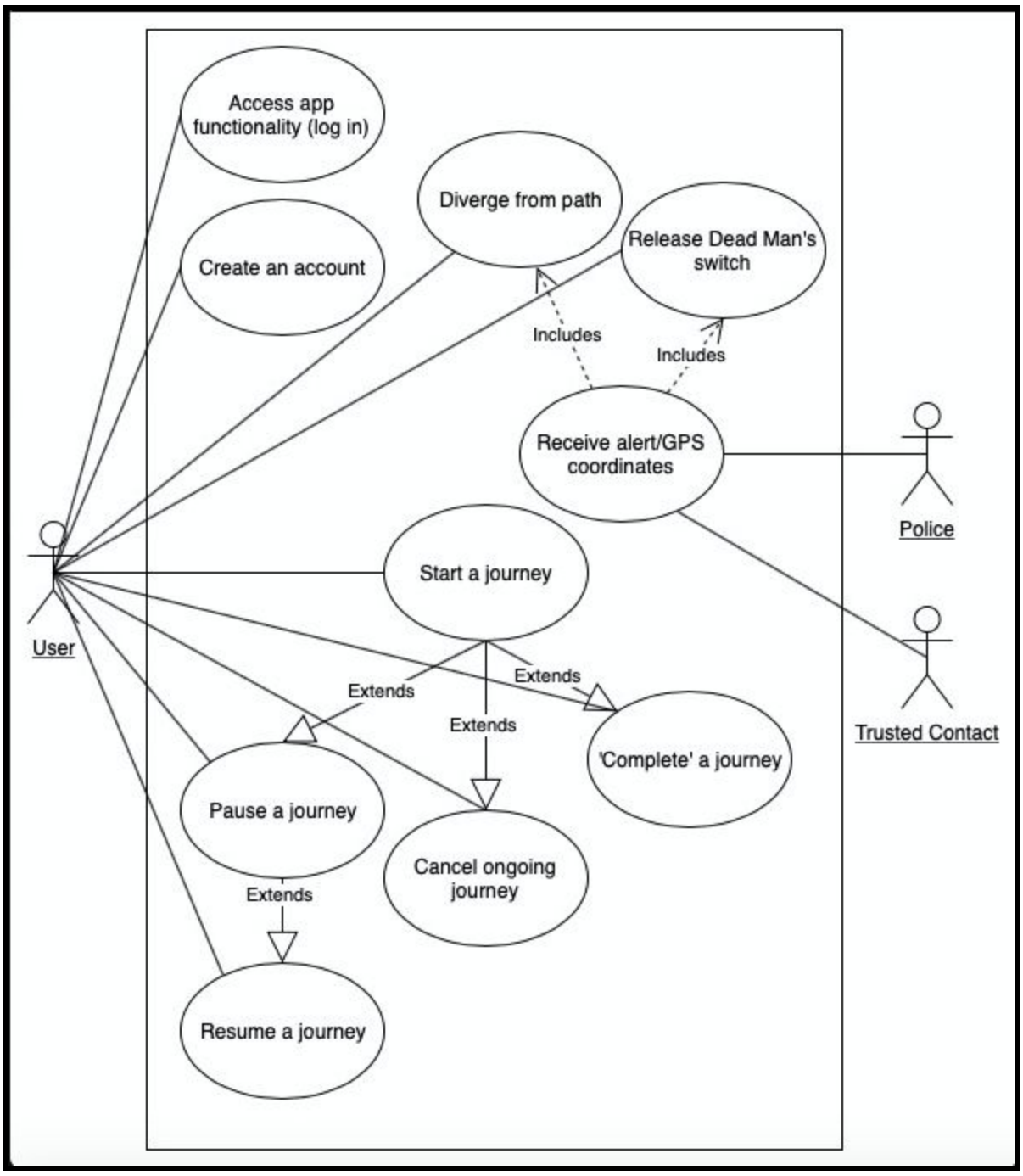


Figure 2—Use Case Diagram

Software Architecture & Data Design

The system consists of the following parts:

Backend:

For our architecture pattern, we decided on a layered pattern since it is commonly used for general desktop applications, like our proof-of-concept . Additionally, it is also the de facto standard for most Java EE applications (Mallawaarachchi 2017).

1. Components are organized into horizontal layers, each performing a specific function within the application. These layers are:
    a. Presentation Layer: User presses a button to view the safety map of an area.
        i. Layer sends request to delegate object, which invokes logics (methods) in business layer.
    b. Business Layer: Fetches user's coordinates via GPS or mobile signal.
        i. Wraps them in a DTO (Data Transfer Object)
        ii. Sends to the server (persistence layer) via HTTPS packets.
    c. Persistence Layer: The DAO (Data Access Object) in this layer might do a little more processing (e.g. to prevent SQL injection) and fire SQL query to the database layer.
    d. Database Layer: The database returns result DAO. Then, it is sent all the way back to the user, traversing these layers.

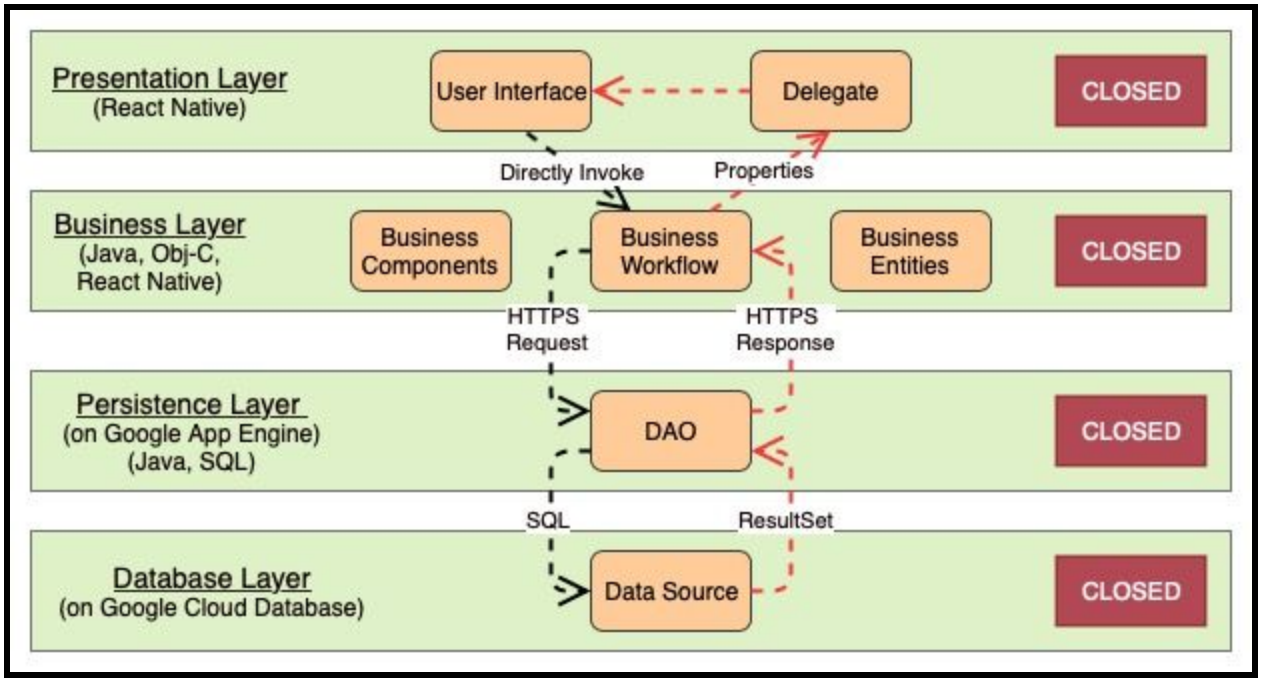


Figure 3—Layered architecture diagram

2. The following is a brief explanation of each component (Meier 2008):
    a. User interface: provides a way for users to interact with application.
    b. Delegate: knows which modules in business layer can process particular request, how to get to module, and data required by it.
    c. Business components: implement business rules, perform business tasks.
    d. Business workflow: define and coordinate long-running, multi-step business processes.
    e. Business entities: actual business entities (products/orders): usually data structures
    f. DAO: execute SQL statements to retrieve corresponding data.
    g. Data source: contains all relevant data.

3. This pattern has its own merits and demerits (Richards 2015 chapter 1); discussed below are two most notable of each:
    a. Ease of development: Relatively easy, since:
        i. Pattern is well known
        ii. Comparably simple to implement.
        iii. Companies designate different skill sets for different layers (presentation, business, database), this pattern becomes a natural choice for most business-application development. (Conway's law (Conway [no date])).
    b. Testability: Convenient to test repeatedly as:
        i. Layers can be mocked/stubbed

1. For example, developer can mock a presentation component or screen to isolate testing within a business component, as well as mock the business layer to test certain screen functionality.

c. Overall agility: While change can be isolated through the layers of isolation feature of this pattern, it is still cumbersome and time-consuming to make changes in this architecture pattern as:
   i. monolithic nature of most implementations
   ii. Change in upper layers causes cascaded changes in lower layers
d. Ease of deployment: can prove troublesome, particularly for larger applications.
   i. One small change to a component can require redeployment of the entire application .
   ii. Thus, deployments need to be scheduled, and executed during off-hours or on weekends, reducing the overall continuity of deployment.

Frontend:

Commonly used in industry, we chose to develop the frontend in React Native since it allows us to integrate the existing heat map, written in Java, without having to rewrite the code. We used React Native with Expo, which acts as “syntactic sugar” for React Native. It gives us easier access to native device functionality such as GPS: hence simplifying overall development process. For example, using Expo, displaying the map and plotting the route requires simpler code .

Following is a summary of our user interface developed using React Native with Expo:

1. The user must first register, then log in to be able to use the application. The password input box hides the entered characters.
2. The ‘Welcome’ page lists all the main functions of the app, namely ‘Start a Route’, ‘Trusted Contacts’, ‘My Account’, ‘Settings’, ‘Report a Problem’, ‘Help’, and ‘Logout’.
   a. The buttons are bright, large and wide-spaced, which ensures ease of use
3. On selecting ‘Start a Route’, the user is taken to the ‘Enter Destination’ page. After filling in the address of their destination, clicking the ‘Add’ button stores the entered details and opens the map screen.
4. The map screen has a dynamic ‘Destination’ label, pinpointing the current destination. The ‘Locate Me’ button focusses the map on user’s current location. Given these two points, the user is then provided with a suggested route (own algorithm used)
   a. This screen also has the “Dead Man’s” Switch’
5. The user can choose to view the crime heat map: this was done by overlaying crime data heatmap on top of the map. Since React Native will use Google Maps on Android and Apple Maps on iOS, it is very helpful that the data (stored in JSON format) could be overlayed onto any map.

6. The app notifies user when they reach their destination—ongoing journey is completed.
7. The user is also able to pause, resume, and/or cancel an ongoing journey.

Database in the server

Database is used for storing data permanently or for a certain period on the server side. It is a much more effective manner storing and querying data, compared to file system. The data stored is mainly in 3 categories:

1. Information of user's account
2. Previous journey's details such as emergency help
3. Graph-based Cardiff map, along with crime data for displaying the levels of danger.

Some data, such as the user's home address, is only stored on client's phone given concerns about privacy, loading speed, and database pricing. Nonetheless, the user has to re-enter these details when they switch phones.

The structure of the database is illustrated below.

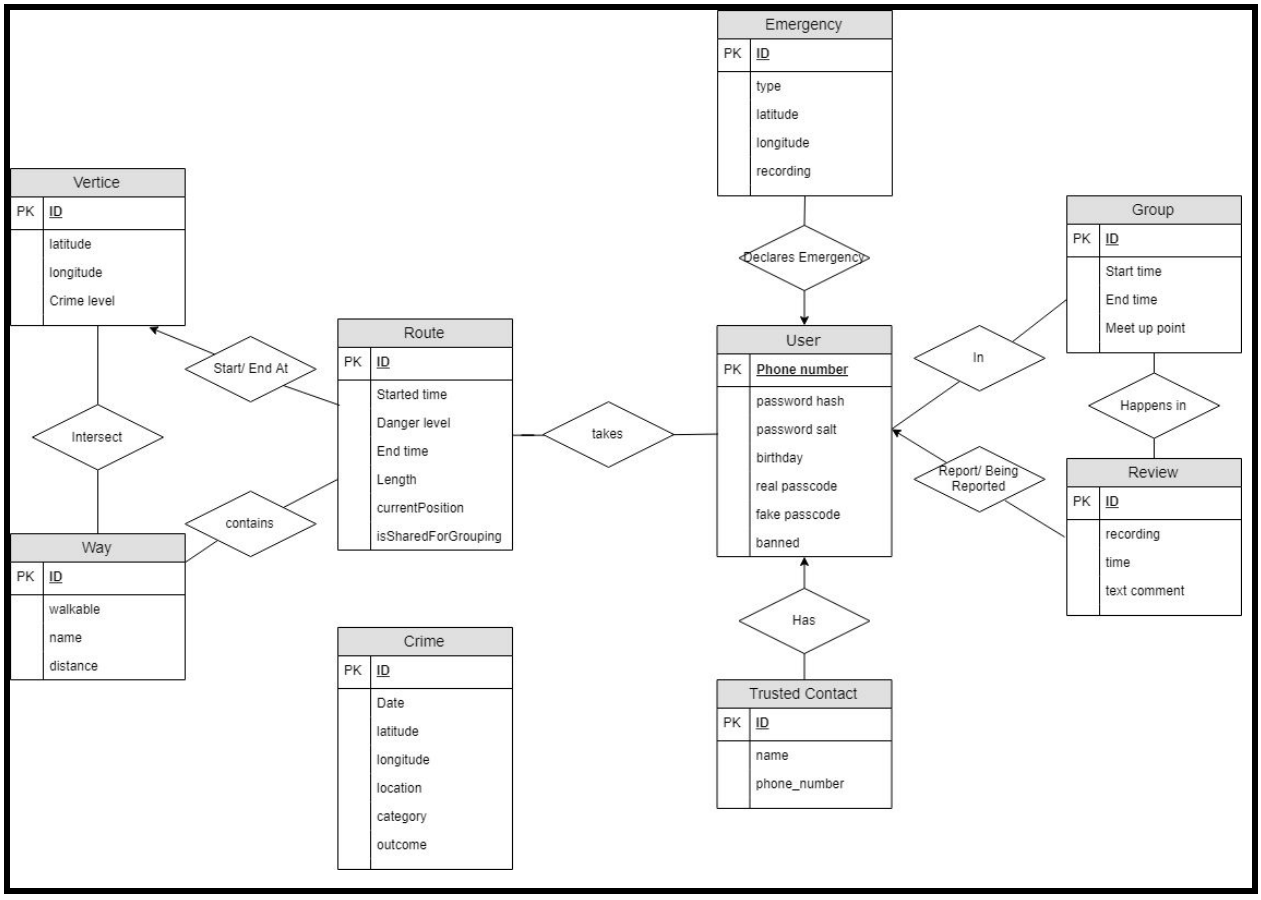


Figure 4—E-R Diagram

Detailed Design

The detailed design demonstrates the plan for the application's features that the team envisions for the client. There were several key factors taken into consideration when designing the application, including (but not limited to):

1. Consistency of layout
    a. All buttons remain in the same place, at the same size across different interfaces.
2. Clear, large buttons.
    a. The application is targeted at users that will be under the influence, so buttons should not be grouped too closely together.
3. Consistent, simple colour scheme
    a. The application should only use a handful of colours to ensure simplicity, and to accommodate as many users as possible.
4. Important buttons should be large and 'loud' coloured.
    a. For example, the dead man's switch is a crucial button in the application. As it is the user's primary button to use when they feel unsafe, it is the largest button in the application and bright red to ensure the user doesn't miss its presence.
5. As little input from the user as possible
    a. (Matanov 2018) shows how requiring as little input from the user as possible provides greater focus on the UI and ease of use for the user. This is implemented by the user already having their starting location loaded into the application based off their GPS when they start a journey. There are also 'toggle' options on some of the settings to minimize the user's input as much as possible.
6. Application name banner on create an account / login and application logo on home screen
    a. Neither are included on other pages to free up room for user experience and interaction.

Create an account

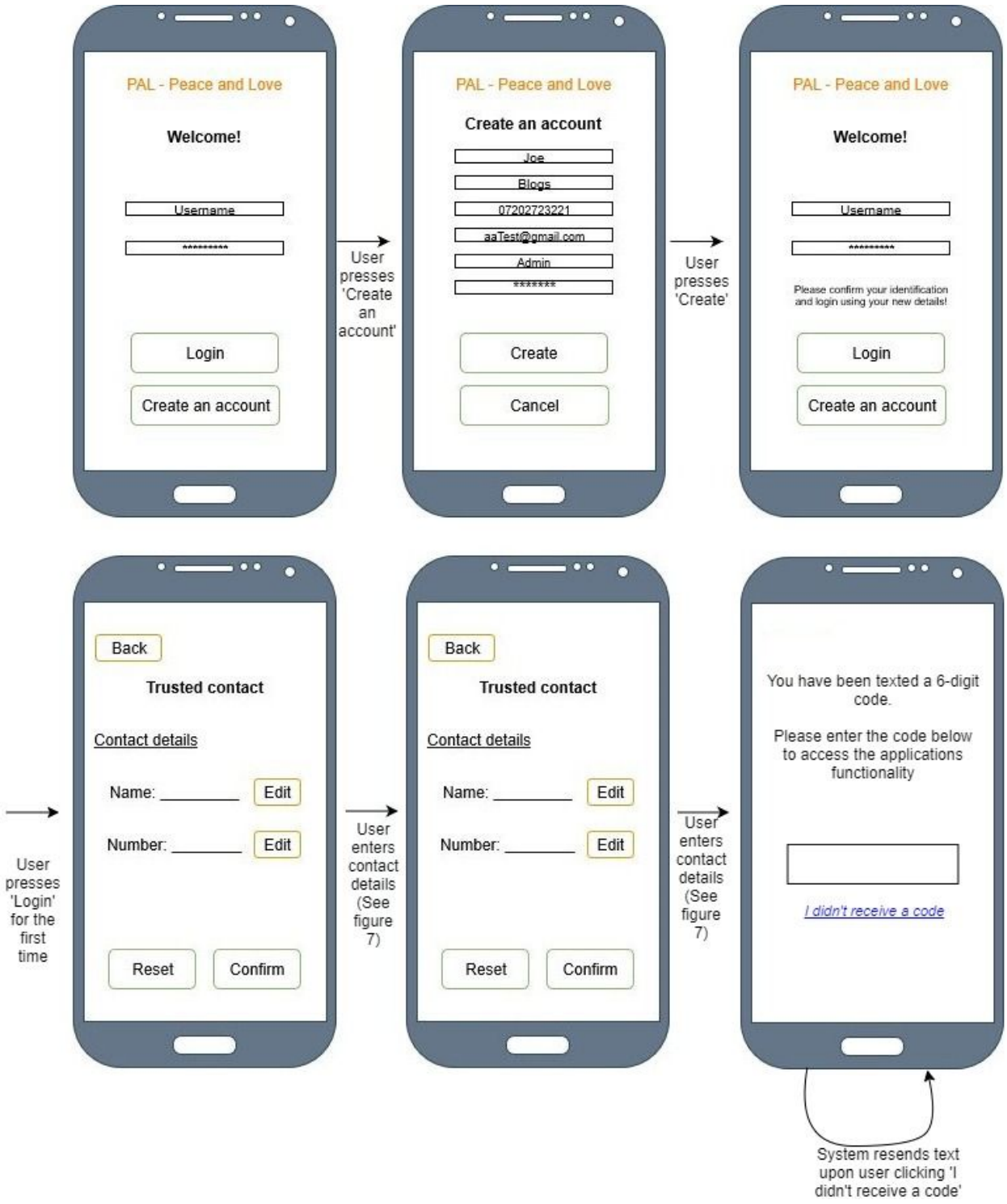


Figure 5—Creating an account

Login

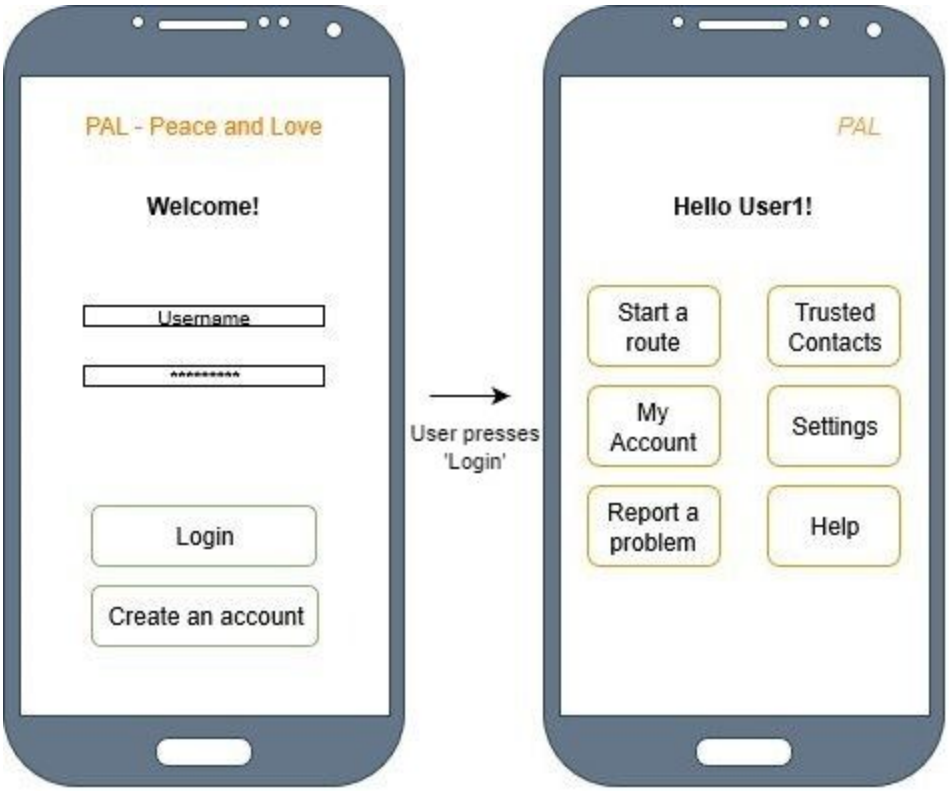


Figure 6—Login

Home page

Figure 7—Home page

Trusted contacts

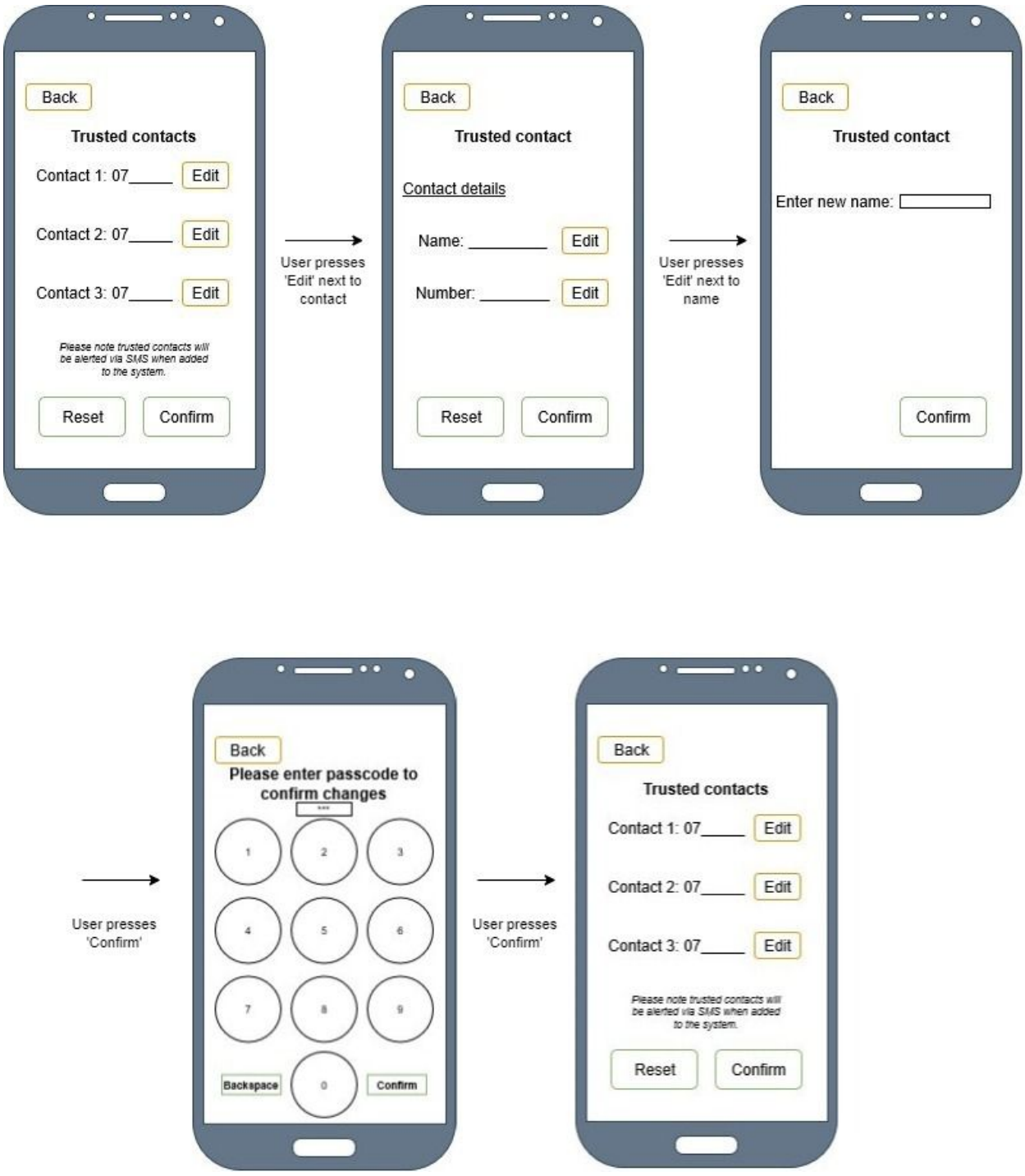


Figure 8—Trusted contacts

Route functionality

Figure 9—Route functionality

My Account

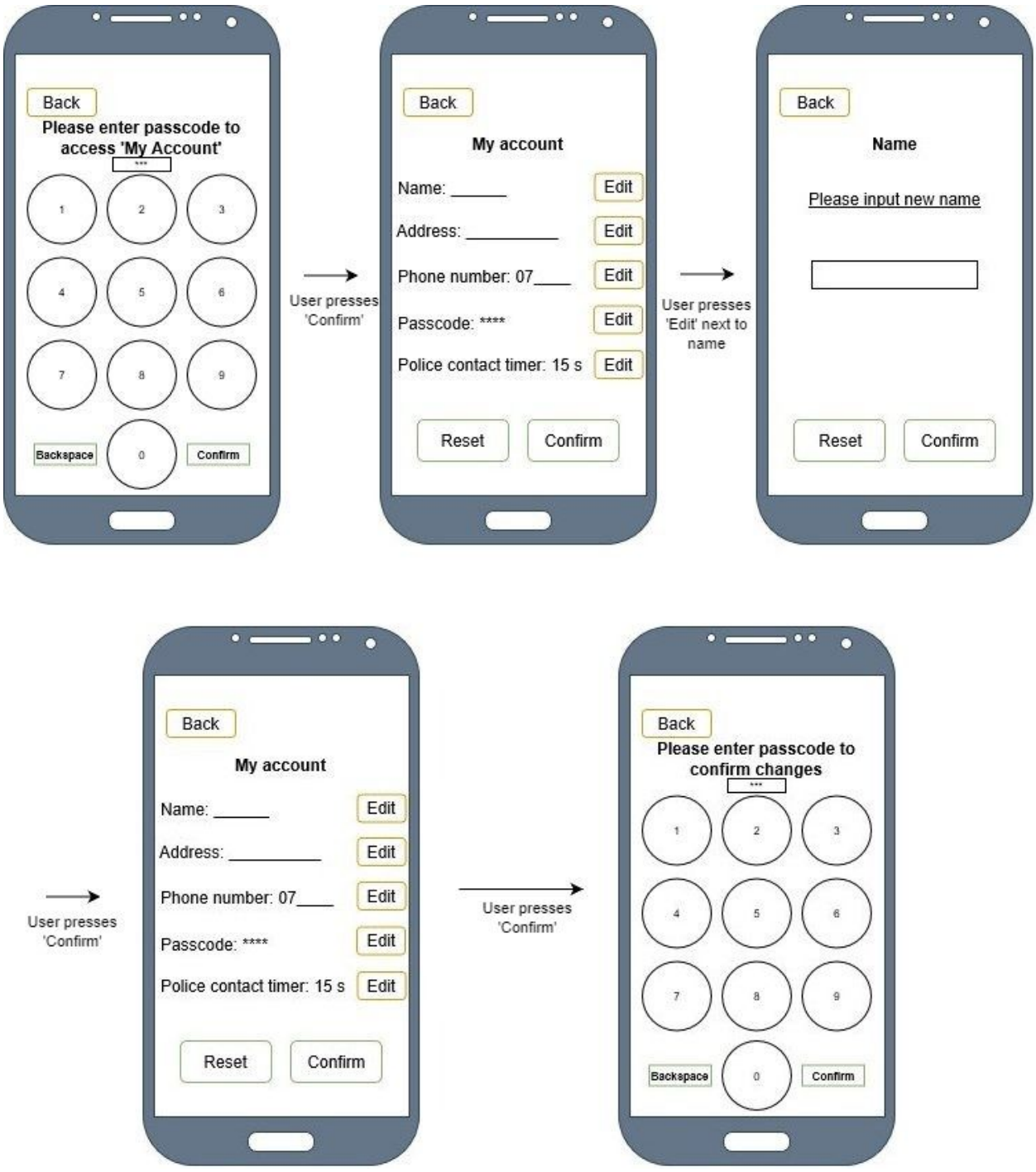


Figure 10—My Account

Settings

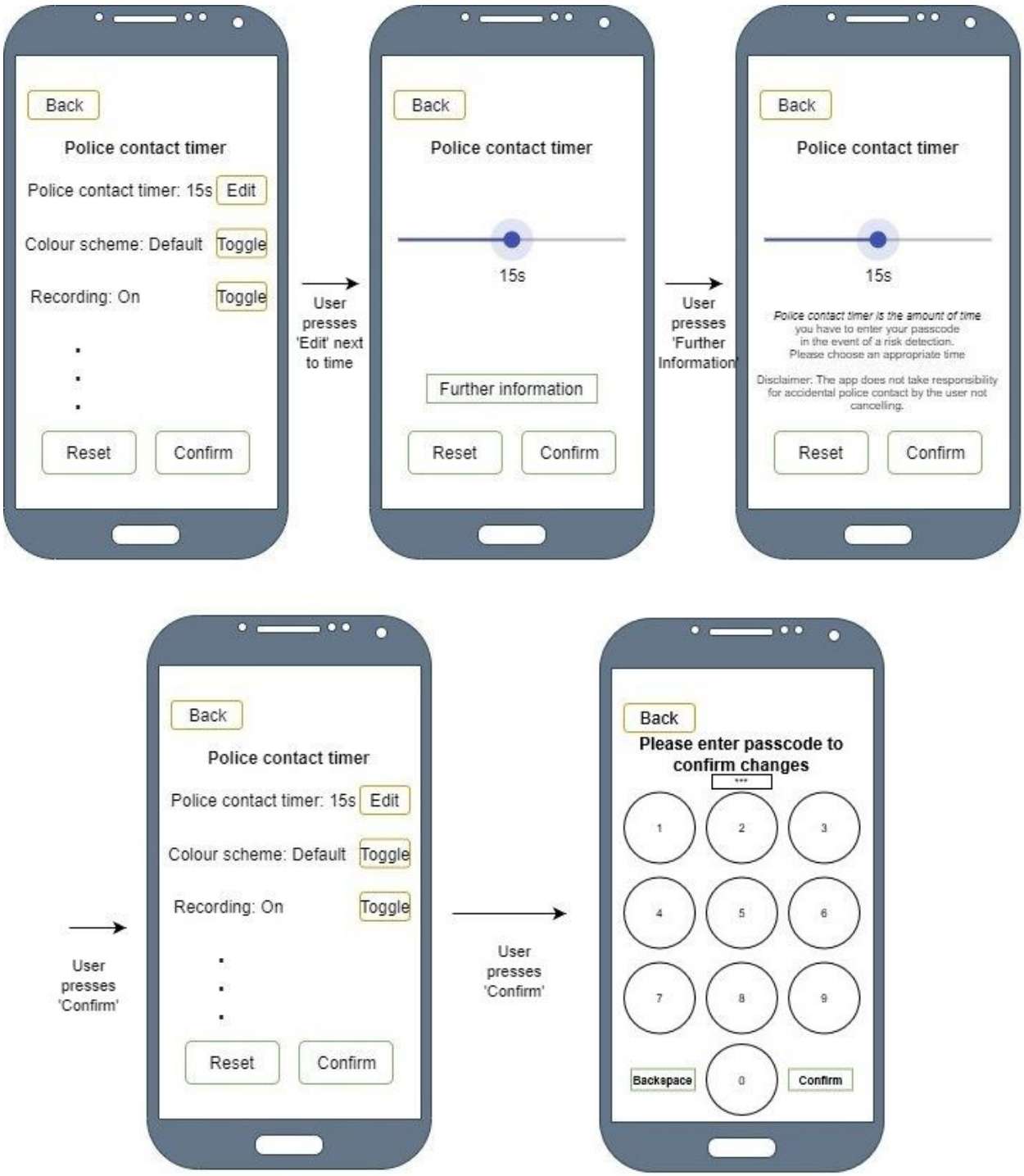


Figure 11—Settings

Report a problem

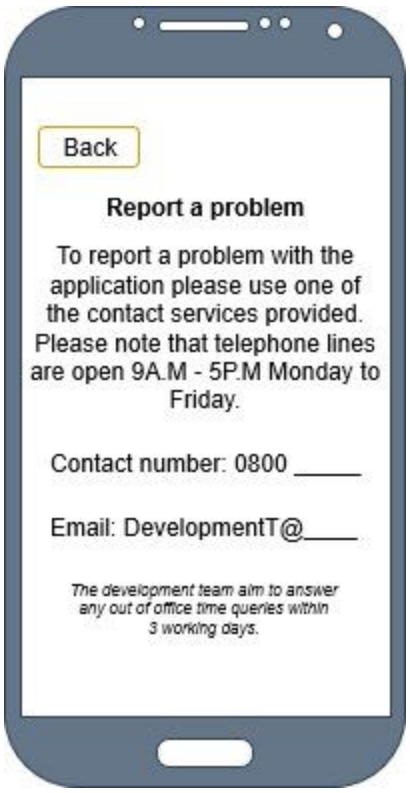


Figure 12—Report a problem

Help

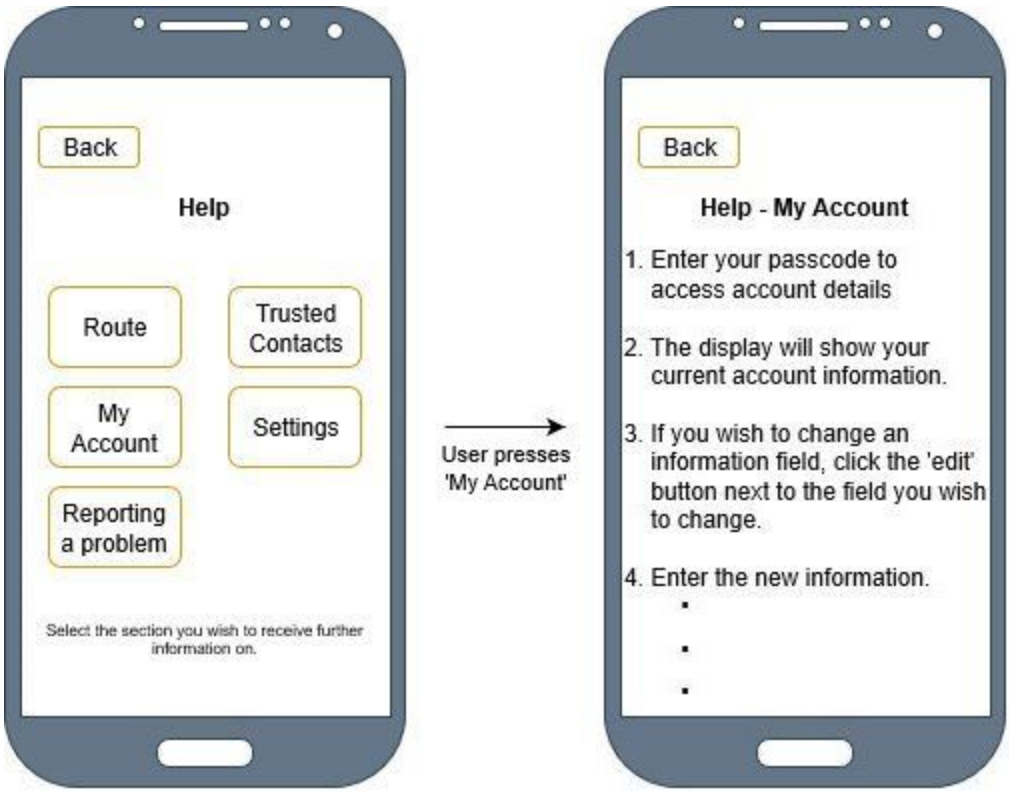


Figure 13—Help

Implementation

Data Structure

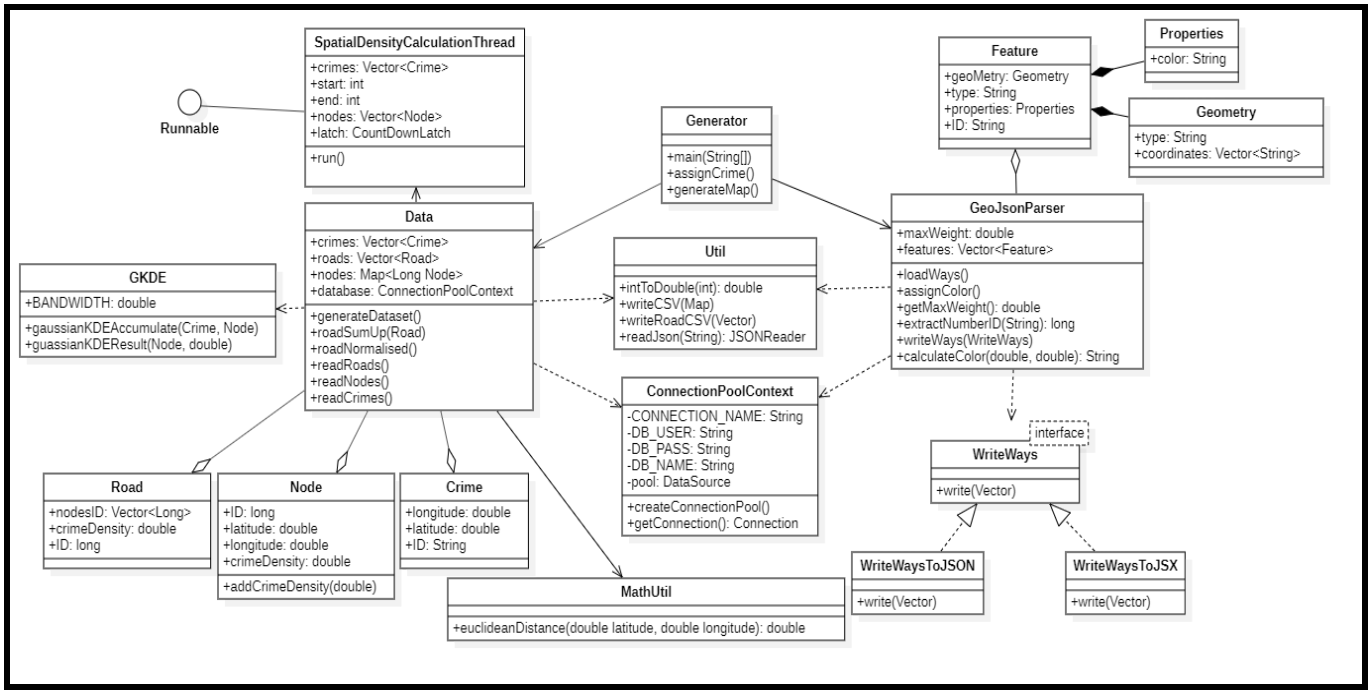


Figure 14—Class Diagram of Server

The data is loaded into the container provided by Java collection library. The container encapsulates useful manipulation for a collection such as sorting.

A table in the database is projected into a class if suitable. For instance, regarding the geographical information, the crime and road instances are stored in the java.util.Vector. This synchronized array collection allows sorting with custom comparator. Moreover, it ensures thread-safety compared to Arraylist and plain arrays. This is important considering the amount of nodes that need parallel processing to fully utilise multi-core processors in modern PCs and servers.

Algorithm

Crime Model

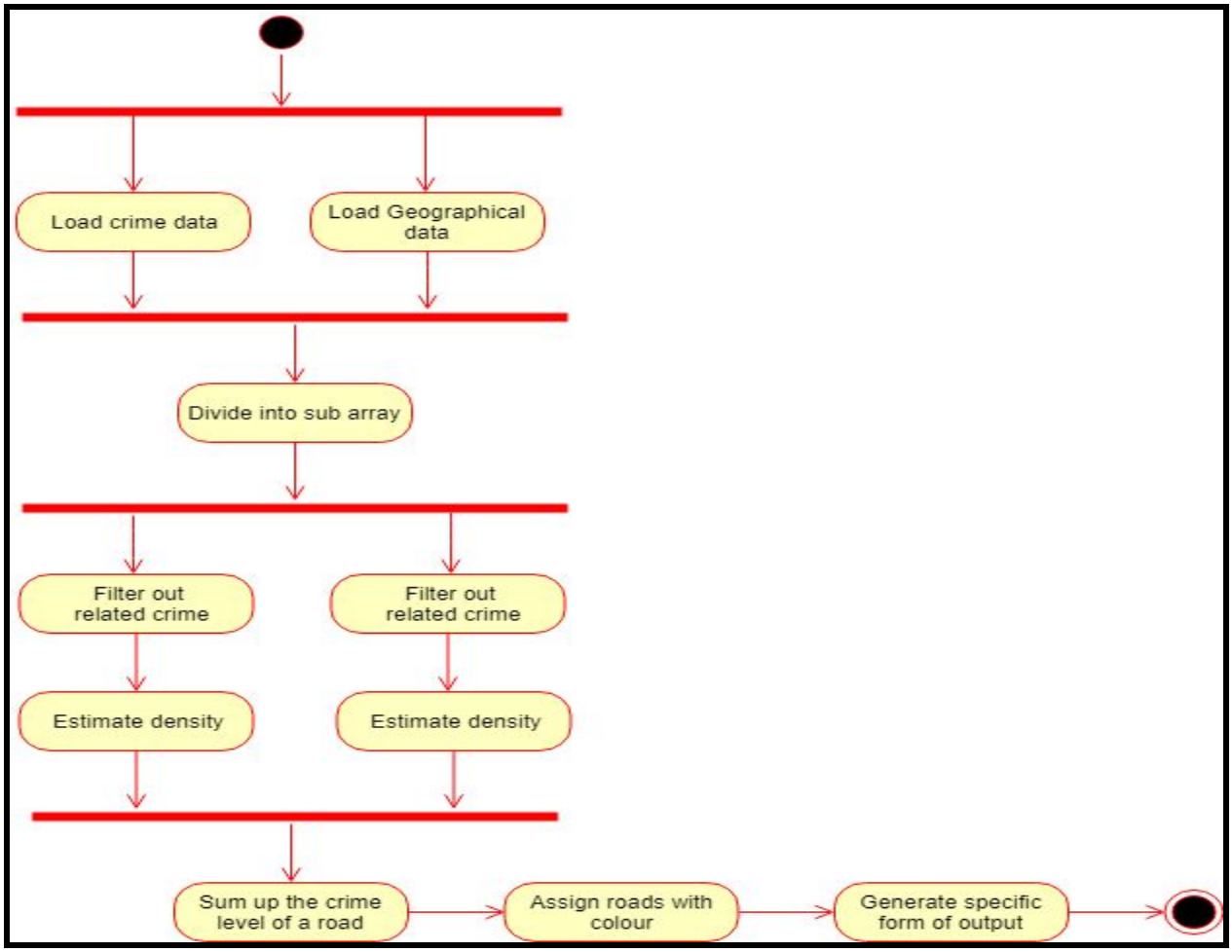


Figure 15—Activity Diagram of the algorithm

The modelling of crime data in geographical distribution mainly implements the paper published by Esther Galbrun et al. (2014).

The crime dataset is available to download on police.uk, which contains basic information of a crime, including geographical coordinates, date happened, and the type of crime. The location is used as a determinant in clustering crimes into nodes. A crime that happens closer to a node will exert more significant influence on it. Depending on the severity of the crime, it's impact on the safety of the route will vary (e.g. shoplifting would carry less weight than assault). This crime data set is not perfect for modelling the safety of a road, mainly because it does not contain the exact time of day the crime occurred. It also misses a lot of unreported crimes, and certain crimes occurring inside buildings may have slight affect the safety on the streets.

Openstreetmap provides a set of vertices and edges. The vertices have coordinates and are an intersection point between roads. The edges represent roads. Hence, we can

approximate the chances of being attacked for a given path by considering the adjacent vertices for which we have calculated a risk value.

For the crime incidents dataset and nodes dataset, the model uses Gaussian kernel density estimation which has advantages compared to statistical hotspot and clustering techniques (Anderson 2009). The □ is the euclidean distance between a point and a crime, and h is the bandwidth.

$$\lambda(s) = \frac{1}{nh^2}\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}}exp(-\frac{d^2}{2h^2})$$

After getting the crime level of all nodes, the path between the two edges would be evaluated to have a risk value of the average of its nodes. This is easy to implement and gives considerably accurate estimation of the risk.

On implementation, the time to run the Gaussian kernel density estimation is proportional to n^2; a plain double-nested loop iterating traversing all these two dataset in Cardiff area costs 149 minutes to complete in an i7-9750h, 8GB RAM device. After refactoring the code with parallel computing and stream filter, the time reduces to around 9.1 minutes.

```
Single-thread processing Finished
Running time: 8940233 ms

Process finished with exit code 0
```

Figure 16—Running time of original implementation

```
Total risk : 1.6653357889932 42E7
Running time:  545307 ms

Process finished with exit code
```

```
Total risk: 1.6653357889932 42E7
Running time: 553894 ms

Process finished with exit code
```

Figure 17—Running time of improved code

Navigation

The navigation can be interpreted as a bi-objective shortest path problem: finding a set of road. There are multiple ways to conquer this problem, such as Dijkstra (Galbrun 2014), Dijkstra-like algorithm (Sedeño-Noda and Raith 2015), and A*. A careful analysis of their outcome and implementation difficulty will be carried out in the next development phrase.

If the performance of newly developed algorithm is not good enough in terms of computational resources and results, a workaround is by using a shortest path finder provided by Google or other map service provider, with tag showing how safe the path is.

Keep code maintainable and readable

Code style guidelines

To improve the consistency and readability of the code, developers determine core coding style principles based on (Google Java Style Guideline 2019) and (Alibaba Java Coding Guideline. 2019). This particularly extracted version cares only the techniques (e.g. frameworks) that this project might use and the most important rules that can improve clarity, in order to minimise the time spent to read through it. The full version of guideline may be found in Appendix D.

Some of the fundamental principles include adding JavaDoc annotation for any method whose function names cannot clearly tell its task, and using org.slf4j.logger for logging facade, which allows developers to plug in the desired logging framework at deployment stage, conducive to the consistency of logging.

Merge Review

To consistently ensure new pieces of code are of high quality and group members can quickly understand its purpose, all submissions in Git are required not to push to master branch when collaborating. Only stable version of code can be merged into master branch. A merge reviewer will ensure that the new code not only free from compile errors but also understandable by reading comments.

User Interface

Flow Chart

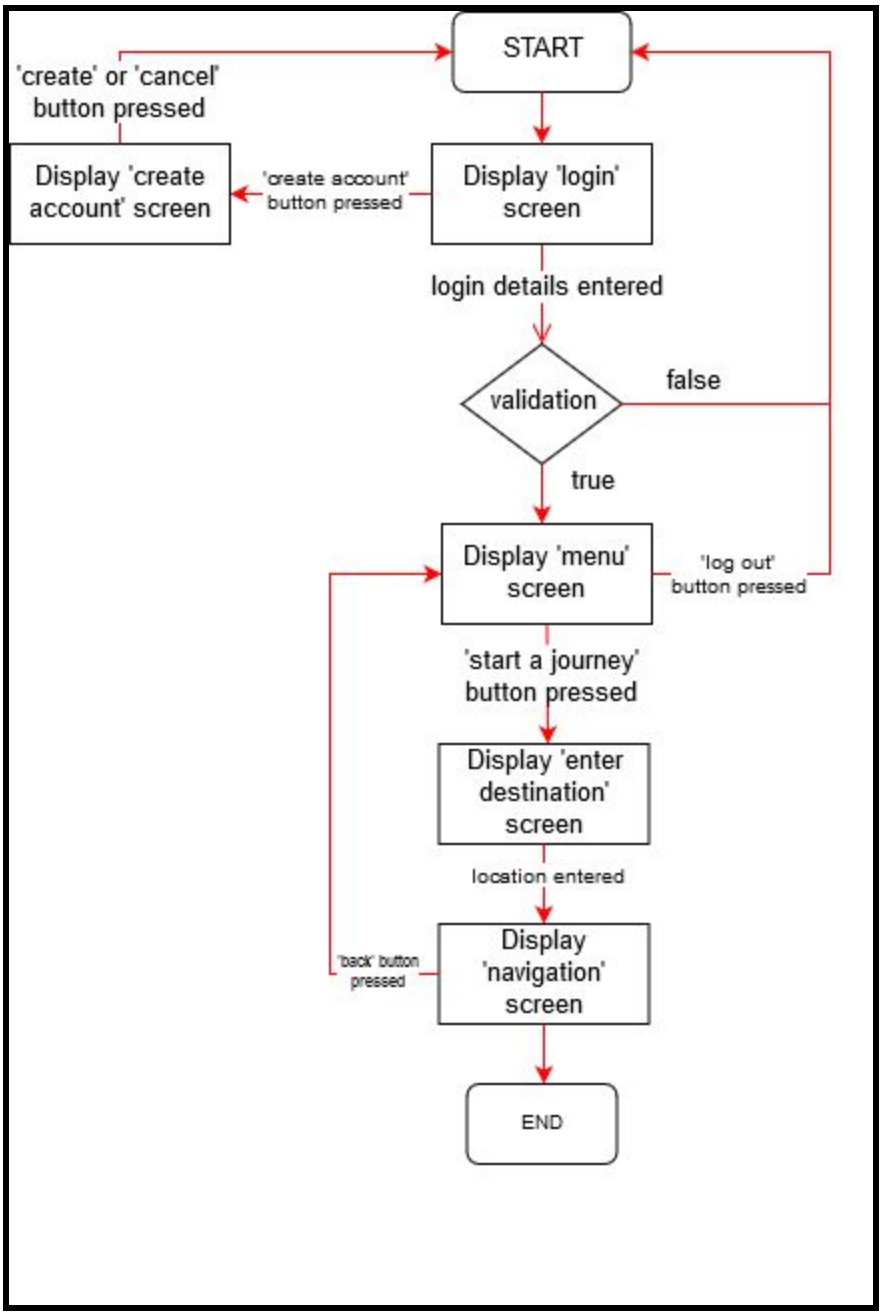
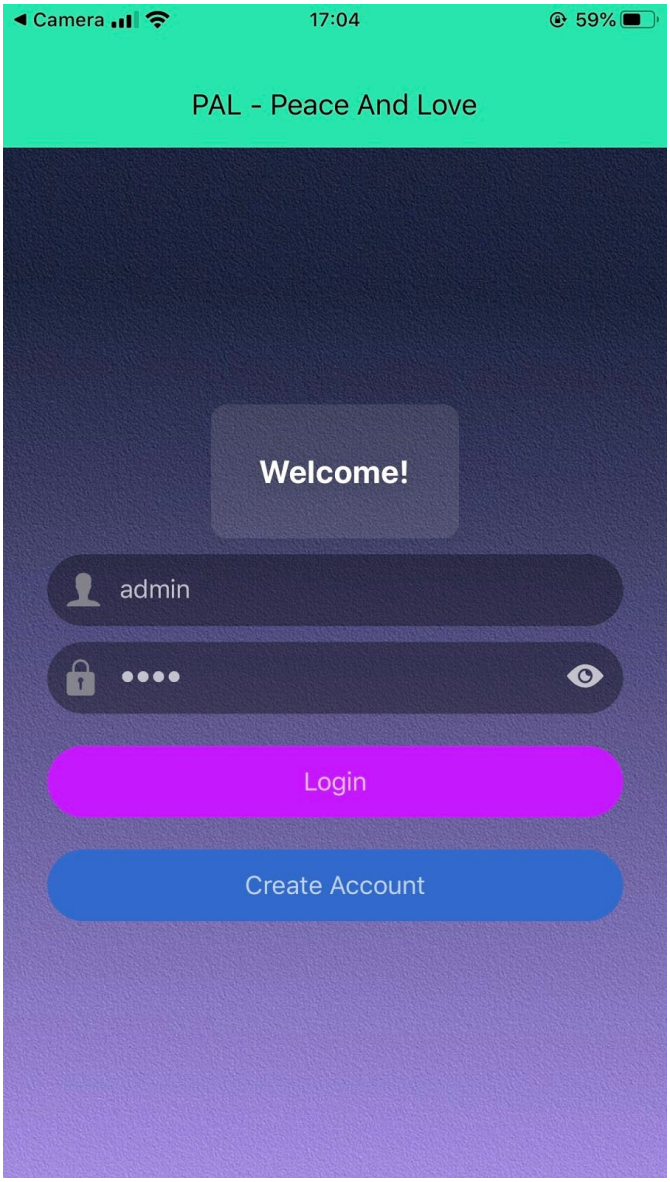


Figure 18 - UI Flow Chart

We developed a working UI in React Native prototype with the following pages:

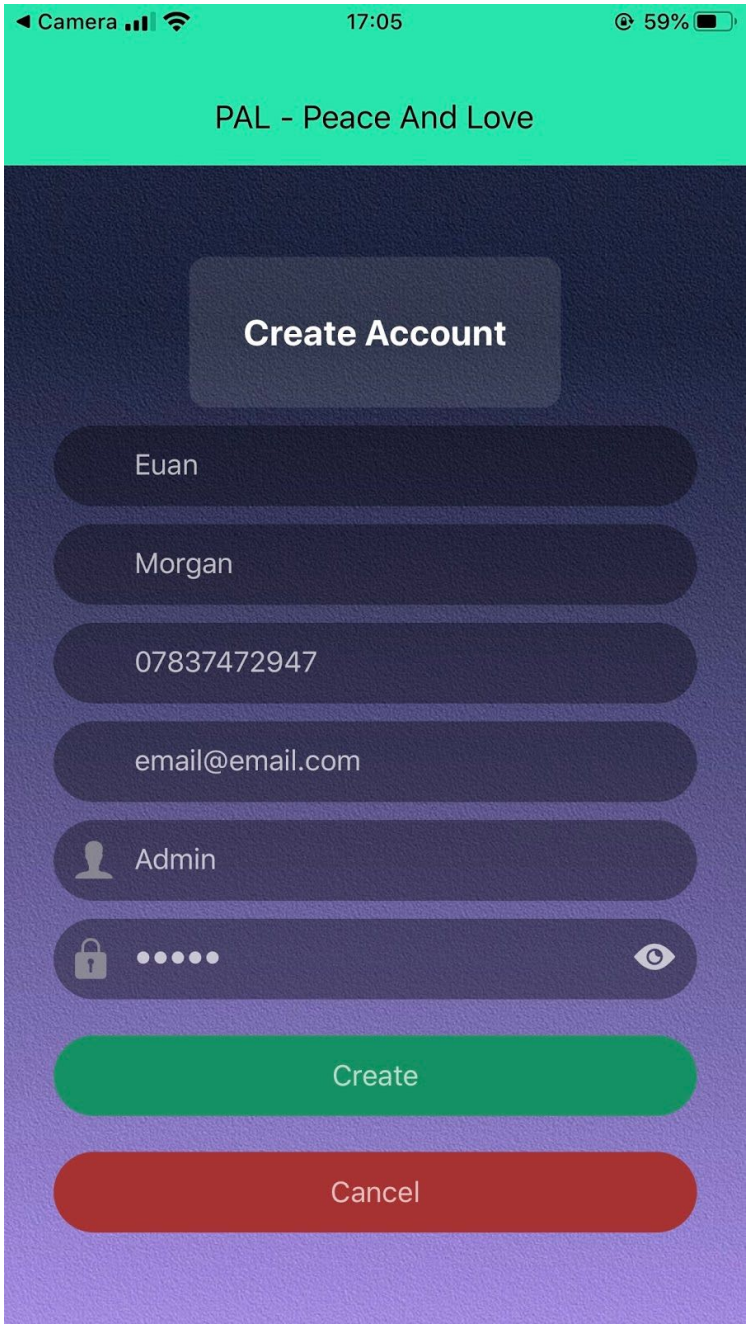
Login Screen



Features:

- Working input fields.
- Password input field obscures characters entered.
- The ‘eye’ button on the password field toggles showing/hiding the password.
- Login button opens the ‘Menu’ page.
- Create account opens the ‘Create Account’ page.

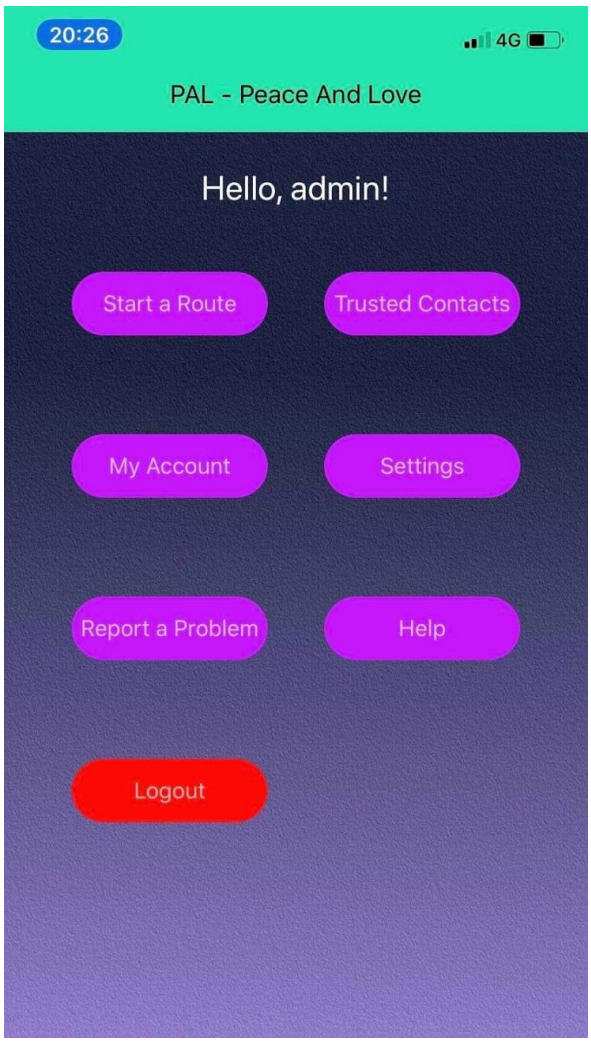
Create Account Screen



Features:

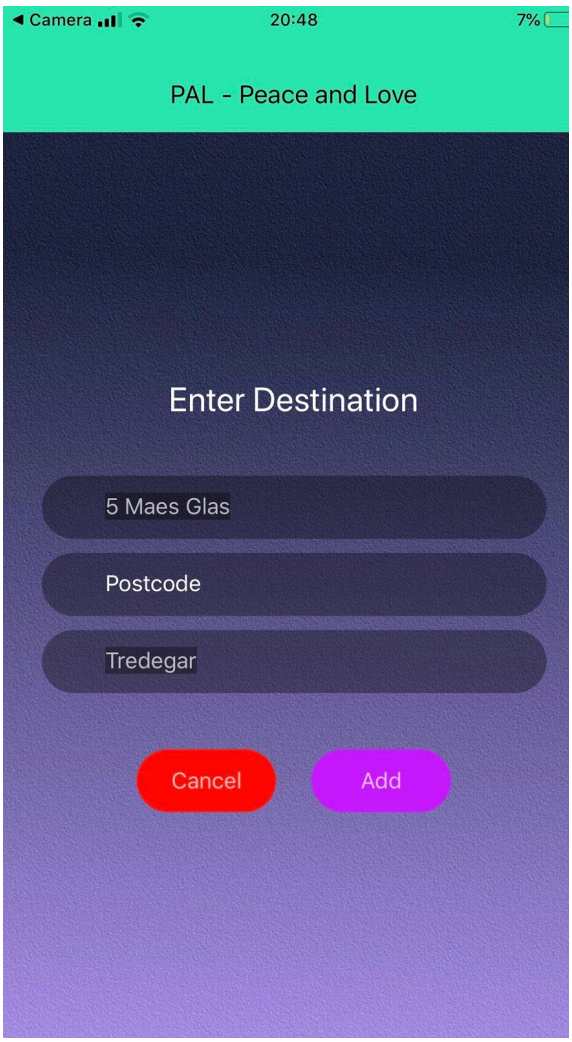
- Working input fields.
- Password input box obscures the entered characters.
- The ‘eye’ button on the password fields toggles showing/hiding the password.
- ‘Create’ button notifies the user that their account has been made successfully, redirects to the login screen.
- ‘Cancel’ button takes the user back to the login screen.

Menu Screen **Set Destination Screen**



Features:

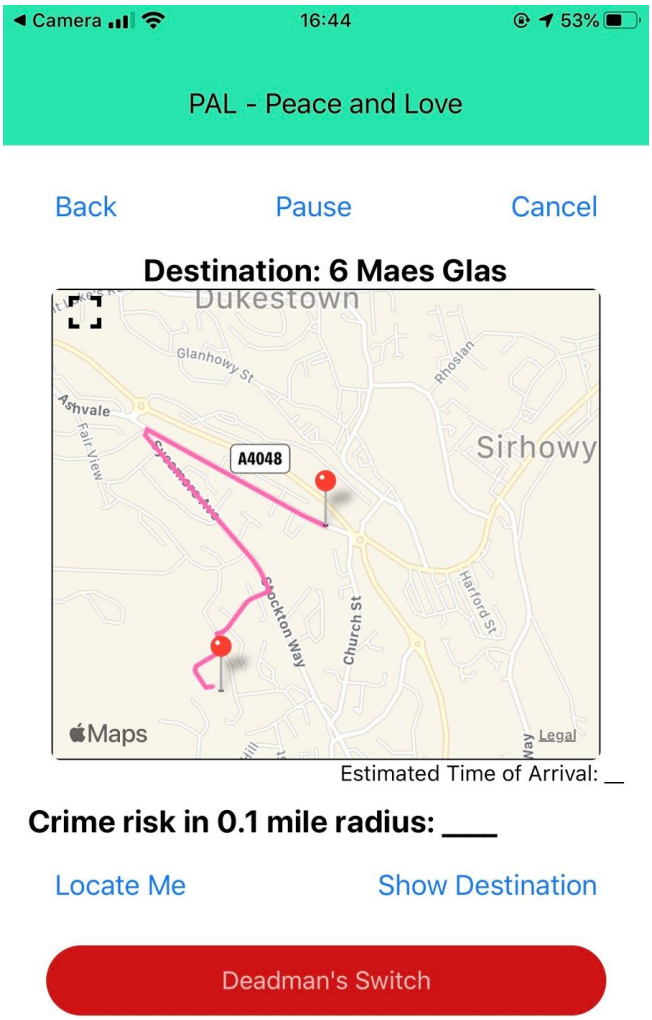
- ‘Start a Route’ button opens the ‘Enter Destination’ page which has a slide-in animation.
- ‘Logout’ button takes the user back to the login screen.
- Dynamic welcome label which displays the username.



Features:

- Working input fields.
- ‘Add’ button stores the entered details and opens the map screen.
- ‘Cancel’ button returns the user to the menu screen.

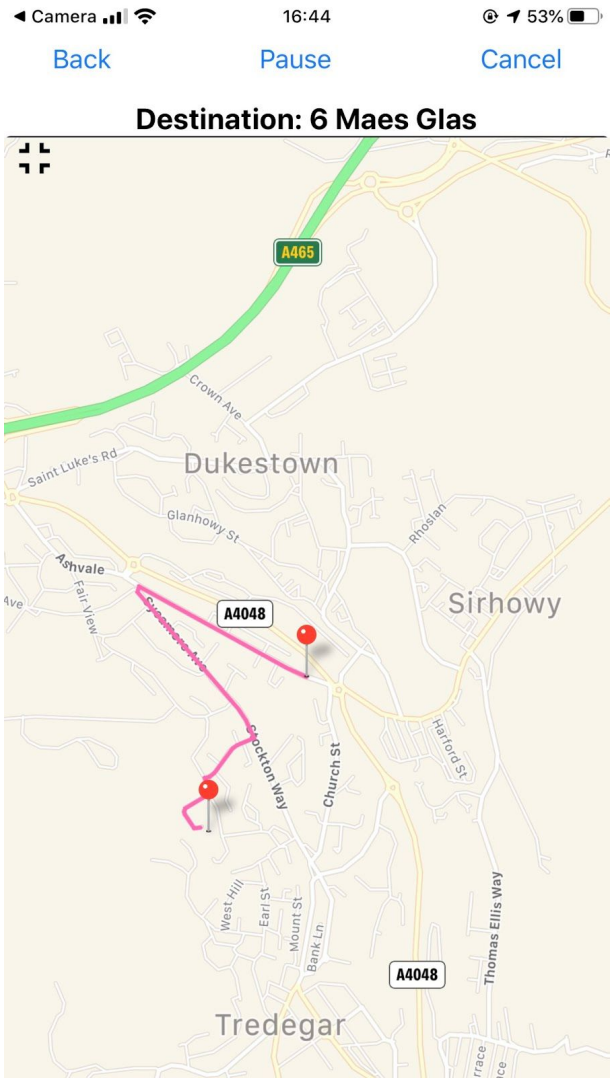
Map Screen



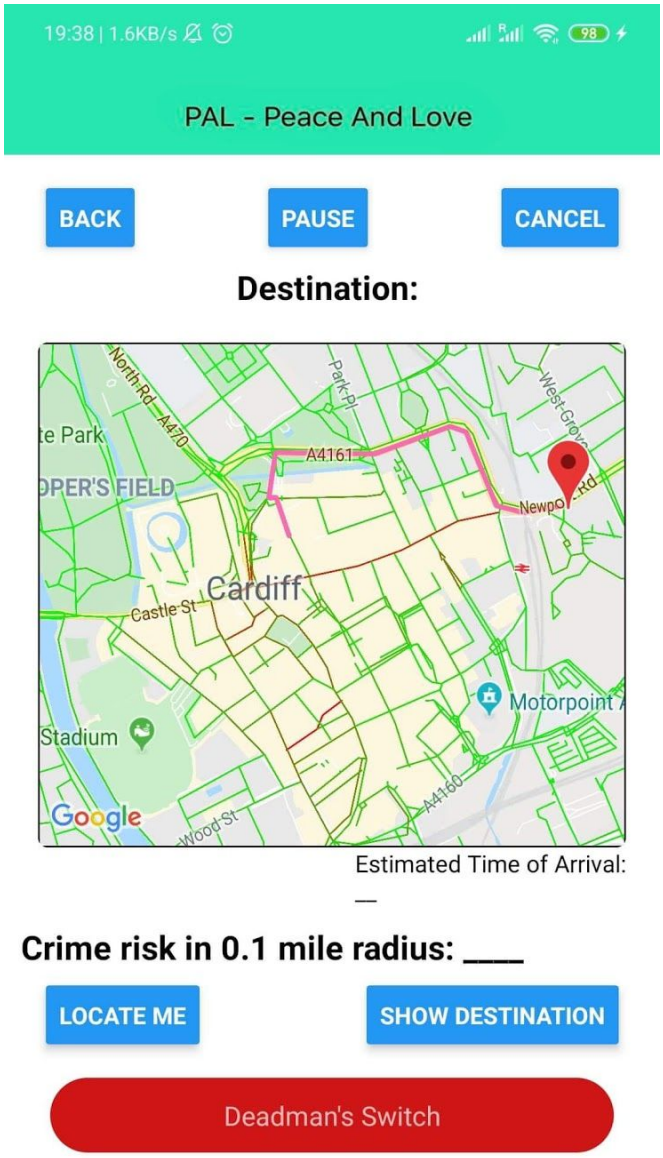
Features:

1. Dynamic 'Destination' label which shows the current destination.
2. Working map:
    a. Uses Google Maps' API to search for a location, locate the current user and provide a route (just to test the maps feature; we will be using our own algorithm to recommend routes).
    b. Marker at destination and current location.
    c. Asks for user's permission to use their location when page loads
3. 'Locate Me' button focuses the map on the current location.
4. 'Show destination' button focuses the map on the destination.
5. 'Dead Man's Switch' button displays an alert that the police are being contacted when user releases it.
6. 'Back' button takes the user back to the menu.
7. 'Expand' icon toggles map size - see screenshot below.

Fullscreen

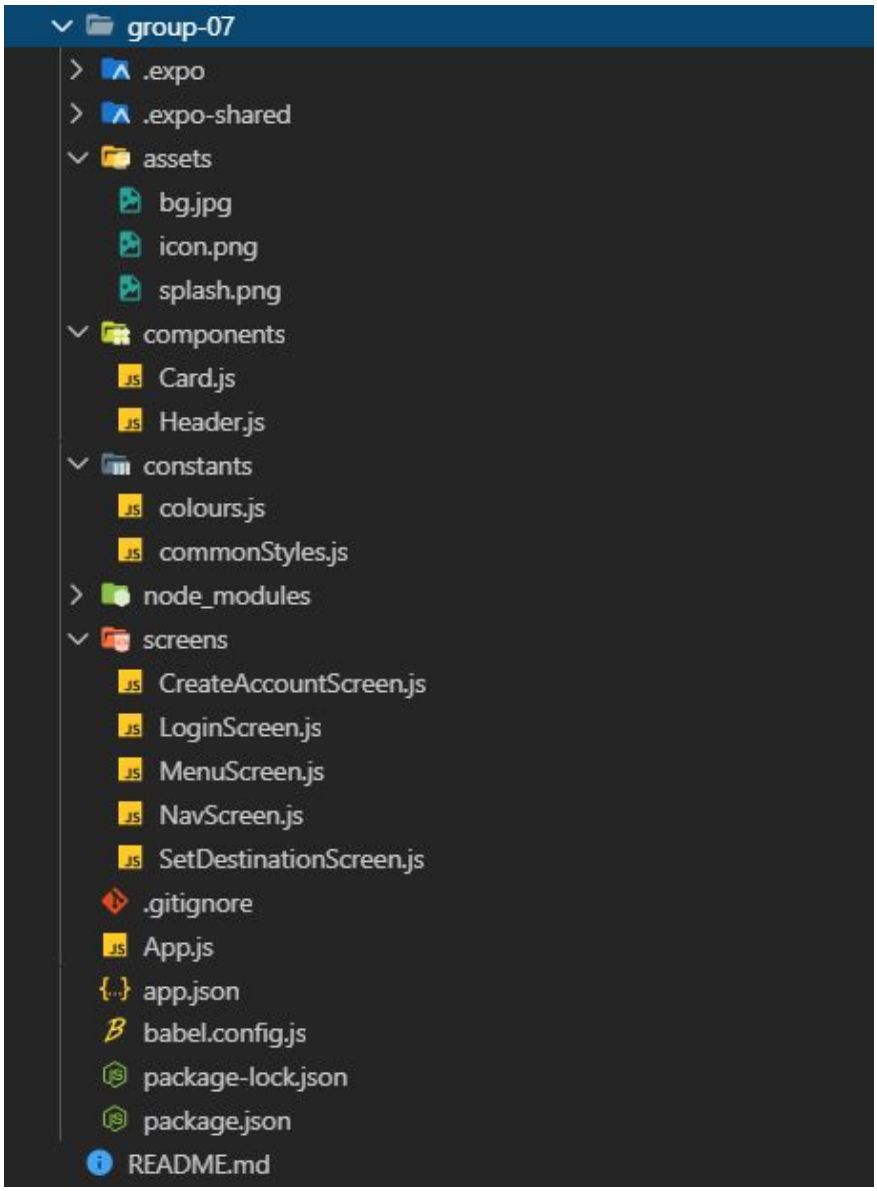


Map Screen with Heat Map



We were able to overlay the Crime Data heatmap on top of the map, and it worked splendidly. The data is stored in JSON format using the longitude and latitude, so it could be overlaid onto any map without having to make platform-specific changes. This is invaluable since React Native will use Google Maps on Android and Apple Maps on iOS. In the above screenshot, red roads indicate roads with a high crime rate, while green roads have a relatively low crime rate. The source code for this will not be included as the values were simply hardcoded into the project as a test.

Directory Structure



The above screenshot shows the directory structure of our prototype. We created separate folders for the assets, components, constants, and the screens which we used. We plan to continue this practice during future development.

The assets folder contains the images used, for example the background of the login and create account screens.

The components folder contains various functions which have been stored in separate files to reduce the amount of redundant code. For example, the screenshots below show

the Header component. Instead of rewriting the code on each page, we instead write it once in its own file and then create an instance of the component where needed.

The header component file:

group-07 > components > Header.js > ...

```
//Imports

import React from "react";
import { View, Text, StyleSheet } from "react-native";

import Colors from "../constants/colors";

/*Main function of the component, currently it returns
the title which is passed to it as an argument and
contains the styling rules specified in the
stylesheet*/
const Header = props => {
  return (
    <View style={styles.header}>
      <Text style={styles.headerTitle}>{props.title}</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  header: {
    width: "100%",
    height: 90,
    paddingTop: 36,
    backgroundColor: Colors.primary,
    alignItems: "center",
    justifyContent: "center"
  },
  headerTitle: {
    color: "black",
    fontSize: 18
  }
});

//Exports the Rendered component
export default Header;

```

Now to use the header, we simply import it:

```
import Header from "./components/Header";
```

And call it:

```
<Header title="PAL - Peace And Love" />
```

This will become indispensable as we create larger and more complex components.

The ‘constants’ folder is where we will store fixed values. Currently we have two files, colour.js and commonStyles.js. The colour file stores the Hex values of colours we wish to use frequently throughout the program. For example, the colour of the Header is stored here so we don’t have to copy and paste the hex value each time we want to use it. In the Header component screenshot, you can see that we import color and then call Color.primary to retrieve the Hex value. The commonStyles file fulfils the same purpose but for the style sheets.

The App consists of multiple screens, so it is vital to have each in their separate file to ensure the code is kept maintainable. Therefore, we created a ‘screens’ folder and each screen has its own file.

We decided on a naming convention to keep the code consistent. The general rules were:

1. Component names should start with a capital letter.
2. Constant names should be in camelcase, starting with a lower-case letter.
3. Screen names should be in camelcase starting with an upper-case letter.

Finally, all code was commented to avoid any confusion. This is beneficial since we are all still in the early stages of learning React Native, and descriptive comments would help solidify our knowledge.

Hence, we maximised efficiency when working collaboratively, and ensured errors were compartmentalised which made debugging much easier.

Testing

Test plan

The most appropriate method of checking our application against the requirements is through manual testing. For this we will use test cases consisting of detailed steps, with the aim of limiting the interpretation of the tester. Since the app's behaviour will vary based on the user's geographical location and the current crime data, it is impossible to give exact instructions for every step. The test cases will be carried out once the functionalities involved have been developed.

Some tests can be automated using Jest. This is a JavaScript testing framework developed by Facebook to work with React Native by default. It will allow us to confirm the code is doing what it should do by comparing the results of execution with prepared snapshots (Facebook [no date]). As these tests depend on the actual implementation rather than the design, any that we make shall be made during development and presented in our final report.

Although React Native applications can be tested on desktop, these test cases will be carried out on an actual android device. This will use an application called Expo which allows the transfer of React Native apps from a development device to a smartphone. This will be done to better allow the testing of native features, mainly GPS. As discussed elsewhere in this report, GPS is only considered accurate to around 5 metres (US Government 2017) under open-sky. The GPS test cases are designed to reflect this.

We will incorporate white and black box tests to maximise our chances of finding bugs early in the development process. Our application uses many independent components: for example, dead man's switch. Each component will then have a unit test which will be written by a different developer to the one who wrote the original code. This will help reduce systematic errors. Lastly, we wrote our test cases before the development of our application to have better architecture, and developer experience.

The individual components that will receive unit tests are:

Function Name	**Test**
Dead man's switch	The button's release function calls enter password function.
Log-in	The function receives the correct data from the user and requests sends a query to the server. Calls login failed or login successful function depending on the result from server.
Server login	looks up and returns the relevant account details.
Get heatmap	Given danger levels of vertices and edges, a color-coded heatmap object is returned.
Display heatmap	Given a heatmap object, it is overlaid on the map.
Gaussian Kernel Density	Given a Crime and Node objects, crime density is added to the node.
Bandwidth	Given the number of nodes and standard deviation, the appropriate bandwidth is calculated.
Calculate shortest path	Given two locations, the shortest path between them is returned.

Following the implementation of individual components, we will test them with a combination of white box and black box tests. Lastly, we will test the system as a whole to ensure it meets the acceptance criteria using the test cases.

The full details of each test plan may be found in Appendix A. Here is a list of all the test cases:

- 1—*The user can create an account.*
- 2—*The user can login with a previously existing account.*
- 3—*The user can successfully start a journey.*
- 4—*The user can finish a journey successfully.*
- 5—*Alarm will be raised if the user significantly diverges from chosen route.*
- 6—*The user can pause and resume a journey that they are on.*
- 7—*The user can cancel a journey that they are on.*

Evaluation Factors

Software Development Model

In the first meeting the team agreed upon using the agile Scrum approach. This was done because all of the team members were familiar with the agile development style. It also allows us to closely monitor the progress we make towards solving the problem, improve our time management and estimation skills based on previous Scrums. Lastly, it allows us to be transparent to our client.
Agile has the downside of not having a clear long-term timescale. Therefore, we don't actually know how much will be implemented by the final deadline.

Initially, we decided to have a Scrum every week and then every other week once our team started coding and had agreed upon relevant standards. This was done to ensure that everyone's work would be consistent; i.e. the app would not be written in two separate frameworks that are incompatible with each other.

Afterwards we allocated tasks that weren't dependent on other tasks, to each team member. Longer sprints allowed for lengthier tasks to be completed but overall progress was harder to measure.

Languages

Our team was informed that previous attempts to implement this brief had failed because too much focus was placed on developing a phone application. This meant that we should prioritize a working prototype. For this reason, we considered Python because of its ease of use, familiarity amongst the team, robustness and popularity, plentiful online documentation and other resources.

The main downside of using Python is that it would limit us to a desktop OS. Most phones do not natively support Python and even if we downloaded a third party application to execute the said Python script, most of them do not support a GUI. There is very little documentation for running Python scripts on phones. Another aspect we took into consideration was the performance. Python is an interpreted language and therefore would be slower than using a compiled language like Java.

Framework

We felt that having a framework which incorporates a high-level language, supports multiple platforms and is being used in industry would be ideal.

There is a large range of frameworks we could choose from. We narrowed it down to using two main Javascript frameworks for our web application, namely Angular and React Native. Angular is maintained and developed by Google whereas React Native is developed by Facebook so they both have a large user base and a lot of documentation available.

Angular mainly uses the strongly typed TypeScript which compiles into standard Javascript. This makes catching errors before compiling easier. It also supports routing and accessing data services out of the box.

React Native applications are fast to build because the framework provides many ready-to-apply components. Furthermore, it supports hot reloading which allows for faster design and code development. They are both based on JS which provides a large package ecosystem.

Overall, they are both very similar in terms of what they provide. The main factor that pushed us towards choosing React Native was that we could implement our heatmap prototype that was used in our initial client presentation without needing to rewrite it.

We agreed on using React Native with Expo since it allowed for easy integration with Google or Apple Maps, depending on whether you're on Android or iOS. This was still compatible with data retrieved via Open Street Maps API—we could overlay the data on the aforementioned maps.

Repository
Next, we decided upon using a remote repository that everyone had access to. We created a repository for a public organisation in GitHub, but this meant that we could only make it public, unless a charge was involved. This is why we chose GitLab in the end.

Google Drive
Collaborating on documents was critical to developing a good presentation and report. The document management system needed to be intuitive, and include features that would prevent people's work from being overwritten, provide an edit history, a commenting feature for cases of contention.

Google Drive provides all of the functionalities mentioned, ensuring that we would all use the same style for the presentation and could easily check what other people had covered to ensure that the presentation is coherent.

Specialisation & Division of Work
Given a large enough project, it becomes increasingly difficult for a single person to understand all of it. This meant that we needed to specialize. The work was split up according to the marking criteria for presentation. This ensured that everyone would be able to answer questions from the client about their part of the presentation and made it easier to write the interim report. We ensured that the team had common knowledge by reviewing everyone's work in team meetings and bringing up topics that we felt needed to discussion, or for which we required clarification from someone working on another section. This allowed for easy time tracking, and verification of work completed. It also helped us identify if people were falling behind.

Strengths & Limitations

React Native
Using React Native to develop the application turned out to be very successful overall. Once we built the front-end, we then integrated the heatmap prototype into the application seamlessly. We were also able to provide the backend functionality of the server using Google Cloud.
However, there were several issues with this approach. Firstly, the installation took a long time due to various issues (e.g. anti-virus software was blocking the installation), and seemed to be dependant on the OS it was running. Android emulator behaved unexpectedly. The heatmap did not work on it, but it worked when run natively on an android phone. While a relatively minor issue, it still took time to troubleshoot. We haven't encountered many issues with our current React Native implementation, but this part of the project is still in its early stages.

Fundamental algorithm
We agreed that making our own bi-objective algorithm is a key part of the project. It is very difficult for the user to work out the safest path even with the aid of a heatmap.

The first step was to provide the algorithm with the right inputs. We needed to find the danger levels on each road; therefore, we used our initial prototype to calculate said risk.
Although it took time, good progress has been made towards a solution. Overall, the team is satisfied with the progress achieved and feels confident going forward.

It is worth keeping in mind that the success of our algorithm cannot be accurately predicted. We will know for certain once the app is released to the public and a large scale study can be done. To further complicate matters, it would need to account for the selection bias since people using our app would probably be younger and more safety conscious.

Time management
The use of Scrum has proved effective. Three meetings a week ensured the team was regularly informed of the overall progress. The social aspects of regular meetings also facilitated seamless communication within a time-restricted environment. Weekly Scrums ensured there was efficient progress for each week of the academic term.

Occasionally, team members could not make a meeting, which created a gap in communication. The team did not stick to the risk assessment technique of video-calling when a meeting was missed: something we plan to improve upon next semester.

Legal, Ethical, Social and Professional Issues

Legal

When considering how to design the app we thought of the legal requirements of the app and how to uphold them. We had to follow the Data Protection Act (DPA 1998) and the General Data Protection Regulations (GDPR 2016). Both DPA and GDPR cover a lot about data protection; however GDPR includes most of the DPA. The two main articles of GDPR that concern us are the "Right to be Forgotten" (Art.13 of GDPR) and "Right to be Informed"(Art.17 of GDPR). Article 13 states that we need to allow for the user's data to be deleted on request. Another important article is the "Right to be Informed" and this requires us to inform the users of every privacy policy we update.

We must inform the users that this app won't stop users from being attacked as it could leave us open to be sued. We have to inform that the app is just a way to find the safest routes and that we are not liable if they get attacked whilst using this app.

We researched the requirements and the rules for the desired distribution platforms. The distribution platforms of choice are Google Play store (Developer Policy Center) and Apple App store (Developer Terms). Both of these platforms have policies to have an app on them however they do overlap a lot. Most of these policies are covered under the GDPR however there are a few others. One of their main policies are to inform the user of what we will be using their data for outside of the terms of service. Another is to not disclose non-public information like personal contact information, or payment details.

Social

When planning this app, we found lots of social issues. The primary issue being that attackers could end up using the app to find victims. We identified two methods to fix this, both with certain merits. The first approach is to only allow people with CRB checks to use the app; hence, only trustworthy users are able to use the app. However, this would significantly reduce the number of users able to use the app. The second method is to use a phone number to verify the user's identification: a more appropriate method, but not as secure. This method allows for the use of Caller Name lookup (CNAM) which takes a phone number and works out the owner's information.

Another interesting social issue is that if people stop taking dangerous routes, at what point does the new safe route become unsafe. This is extrapolated from a theory called Displacement Theory (NSW Attorney General's Department 2011), which suggests crime-prevention measures don't prevent crimes, they simply relocate the crime.

Ethical

We must specify our responsibilities concerning users and their safety. It is imperative that we elucidate the fact that the app only suggests safest routes based off reported crimes; however, a large proportion of crimes go unreported and thus, crimes can happen

anywhere. We must clarify that their phone will only be tracked if the app is running or during an active route. Since GPS tracking can be inaccurate, it's vital we warn the users to not follow the map blindly.

Professional
We decided that the app's features should focus on increasing users' safety, rather than their vulnerability. It's essential to have a system which reacts to bugs quickly, and alerts users of any bugs that might put them at risk. Hence, a 'Report Bugs' feature is vital. Once the prototype is completed, we will need to create a team to maintain the app and address any problems that arise.

Work Plan

We agreed that using Agile would be the most efficient way of delivering the software to our client. To further encourage communication within the development team, we chose a Scrum approach - although for practical reasons, we only do our "stand-ups" 3 times per week. We felt this would be less exhaustive and more appropriate than doing them daily because we aren't working on the project full-time. The diagram below better illustrates how Scrum works.

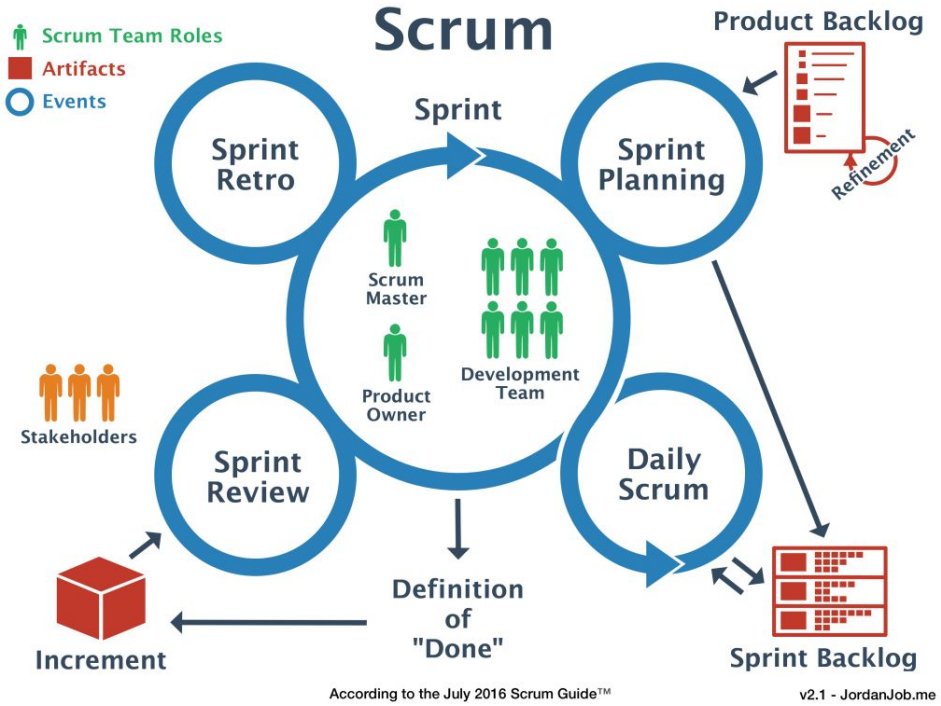


Figure 19 - Scrum Development
(James Wu 2019)

Milestones

Usually we would not include a Gantt chart while developing in Agile, mainly because it is difficult to produce an accurate time plan. But to communicate our milestones clearly, we have decided that it was appropriate in this case (after consulting with the module leader).

Illustrated below are the separate phases of the project indicated by varying colours. Because of tasks varying on a weekly basis, it demands us to be an adaptable team. Currently, we have completed the planning stage and we are working on the design stage. We have split up the tasks in the project to facilitate multiple members actively working on the project simultaneously and allocated the tasks according to people's skills, experience and preferences. Since this has proven to work well thus far, we will continue to work accordingly. Our estimations for the completion of different phases are dependant on a number of variables, so we have allowed time for errors to be fixed at each phase of the project. Below, you can see that for each phase there is a "preparation" or a "conclusion" task, these are to allow members of the group that have struggled to reach the target in time to catch up, during these periods, all team members are expected to help their fellow team mates reach the milestone in time.

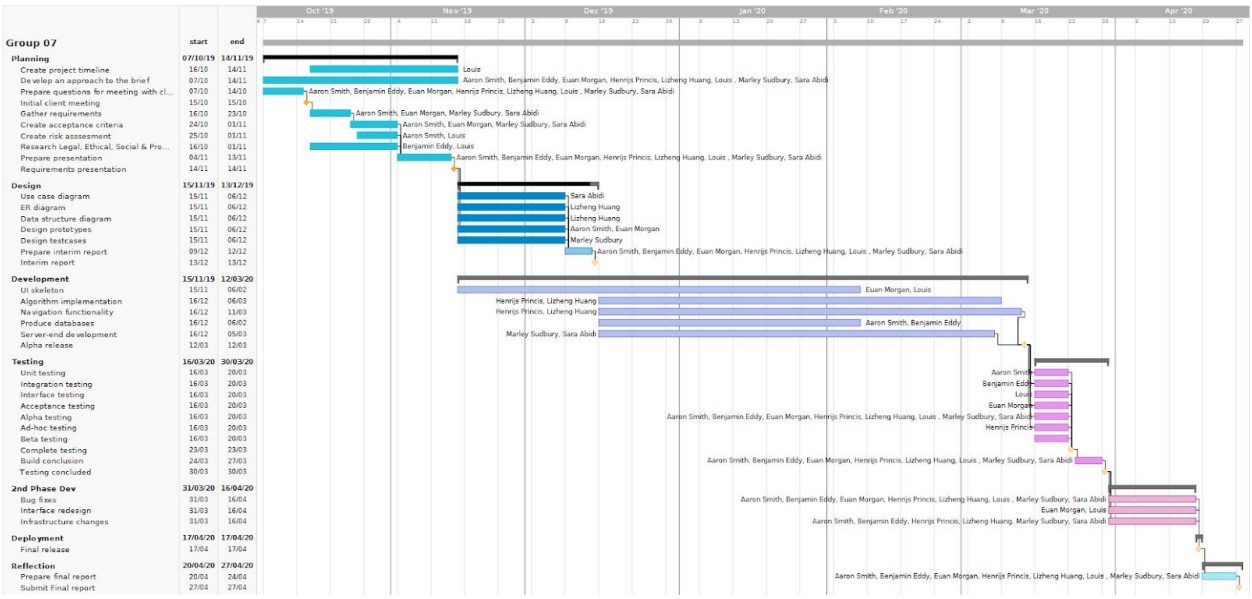


Figure 20 - Gantt chart

Risk Assessment

Assessment of the team's performance

Risk	Likelihood	Impact	Strategy to minimise disruption
Team members communicating inefficiently (1)	Medium	Large	Team members must update the group about tasks that they have completed, and tasks they plan to work on during the Scrum meetings
Team members miss meetings (2)	Medium	Moderate	All team members are aware that Monday's meeting at 12:10pm is compulsory. This meeting must always be attended by all members, unless there is a good reason for absence . Attendance of meetings at 13:00pm on Tuesdays' as well as 10:00am Thursdays will also be monitored by the rest of the team. If a team member cannot attend Tuesday or Thursday meetings, they're welcome to video call the team members and actively contribute. It is expected of all team members to communicate their absence, before the meeting commences.
Loss of source code (3)	Low	Large	Multiple team members will be responsible for a physical back-up of the project's code on devices such as a laptop or an external hard drive. All updated code will also be uploaded to a private GitLab repository
The team has a disagreement which affects the overall productivity (4)	High	Moderate	Naturally, disagreements will occur within the team. In order to minimise disruption all team members have agreed to approach disagreements maturely, such that their relationship with teammates remains unaffected.

Assessment of the system's performance

Risk	**Likelihood**	**Impact**	**Strategy to minimise disruption**
The user has a disability (A)	Medium	Large	The system will have a 'colour-blind' setting where the colour contrasts of the heatmaps can be alternated to accommodate users that may suffer from colour-blindness. The text on the application will be as minimal as possible, as a result, we will ensure that text has a minimum font size 12, to minimise the possibility of users with eye defects experiencing problems when using our application.
The application does not track the user's location accurately (B)	Low	Large	The OpenStreetMap's API will be rigorously tested by all 8 members of the development team on separate devices to measure its accuracy. The performance statistics will then be provided to the user upon delivering the application; ensuring that they are aware of the reliability of our system.
The app fails to alert the police when an attack / diversion takes place (C)	Low	Large	We will also repeatedly test these features on separate devices and an efficiency percentage will also be acquired from these results. We will provide this statistic to the user, as well as the rest of the performance statistics.
The app does not run on a different devices (D)	Low	Moderate	Although the brief mentions that the app should be developed for Android, we have decided to use the React Native framework due to its ease of cross-platform implementation (both Android & iOS).

Template (Phillips, H 2018)

Risk Map

		Likelihood of occurrence		
		High	Medium	Low
Potential scale of impact	Large		1, A	3, B, C
	Moderate	4	2	D
	Small			

Template (Phillips, H 2018)

Evidence

We decided to use Google Docs and Gitlab to collaboarate on each aspect of the project, ensuring that each component integrates successfully.
Rationally, one collaborative project broken down into smaller components working cohesively seemed better than several separate, poorly consolidated files.

Crime Data Heatmap
During the first week, Lizheng demonstrated his prototype for analysing crime data (from the South Wales police) and displaying it on a heat map. Afterwards, we discussed potential ways to integrate this with our future code. Lizheng created a GitLab repository to allow us to view and understand his prototype.

Heatmap GitLab repository

Lizheng Huang › Krapp-Heatmap › Details

K

Krapp-Heatmap

Project ID: 7241 | Leave project

Star 0 | Fork 0 | Clone

12 Commits | **2** Branches | **0** Tags | **2.3 MB** Files

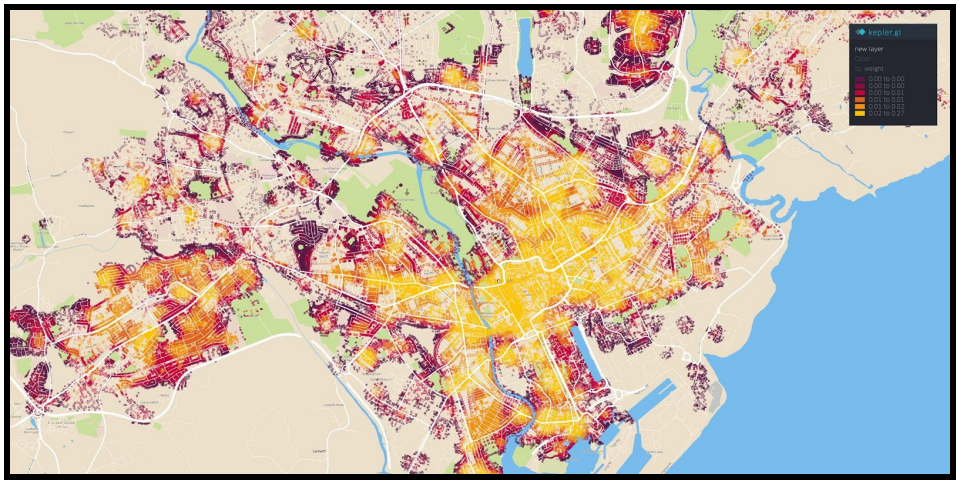
master | krapp-heatmap / | + | History | Find file | Web IDE

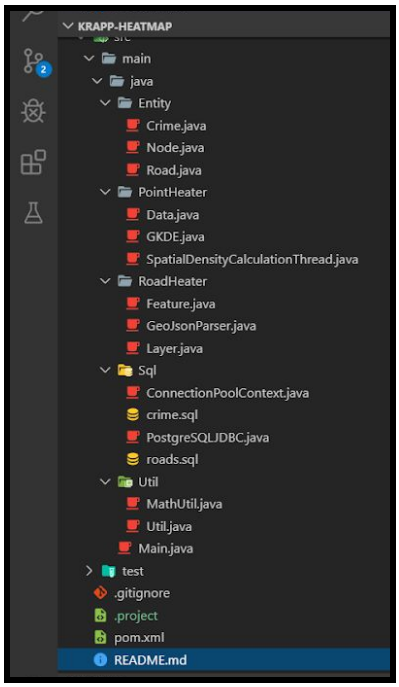
Readme.md updated for the code change
Lizheng Huang authored 1 week ago
530c8c29

README | No license. All rights reserved

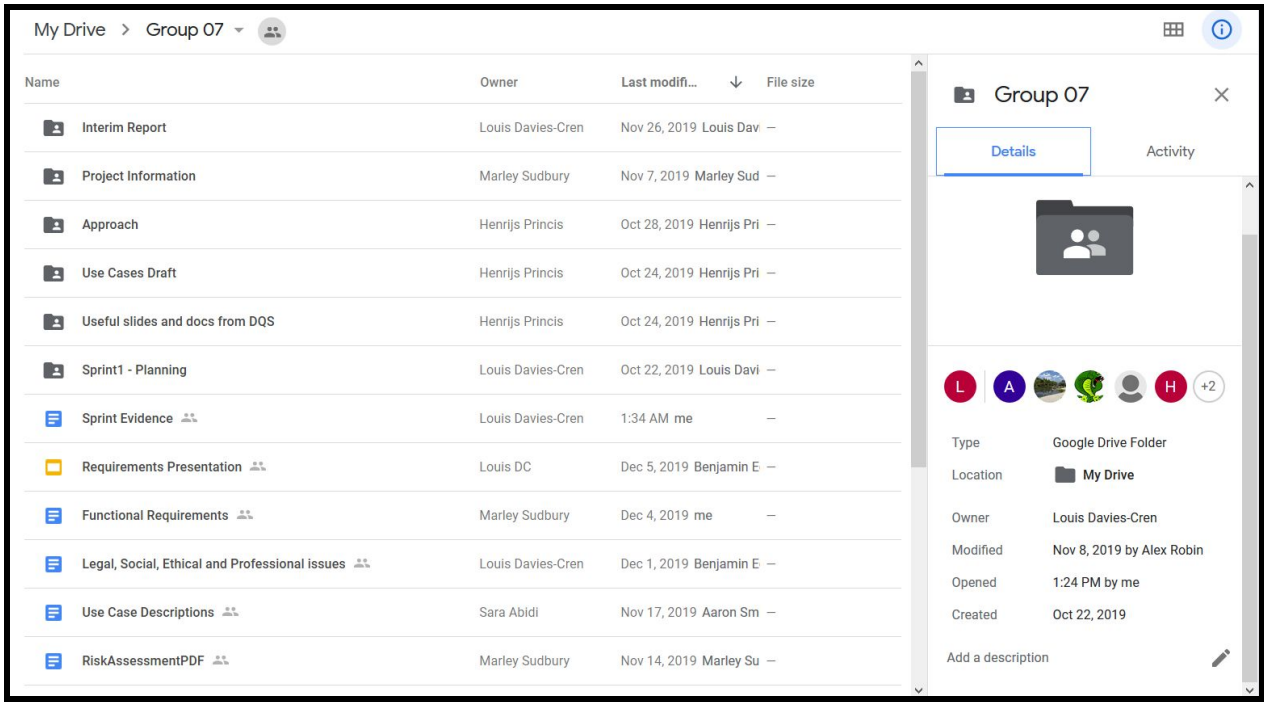
Name	Last commit	Last update
Test	Annotation and TODO added	1 month ago
resource	Annotation and TODO added	1 month ago
src/main/java	Maven framework added	1 week ago
.gitignore	Road heatmap added	1 month ago
README.md	Readme.md updated for the code change	1 week ago
pom.xml	Maven framework added	1 week ago

Crime data heatmap of Cardiff





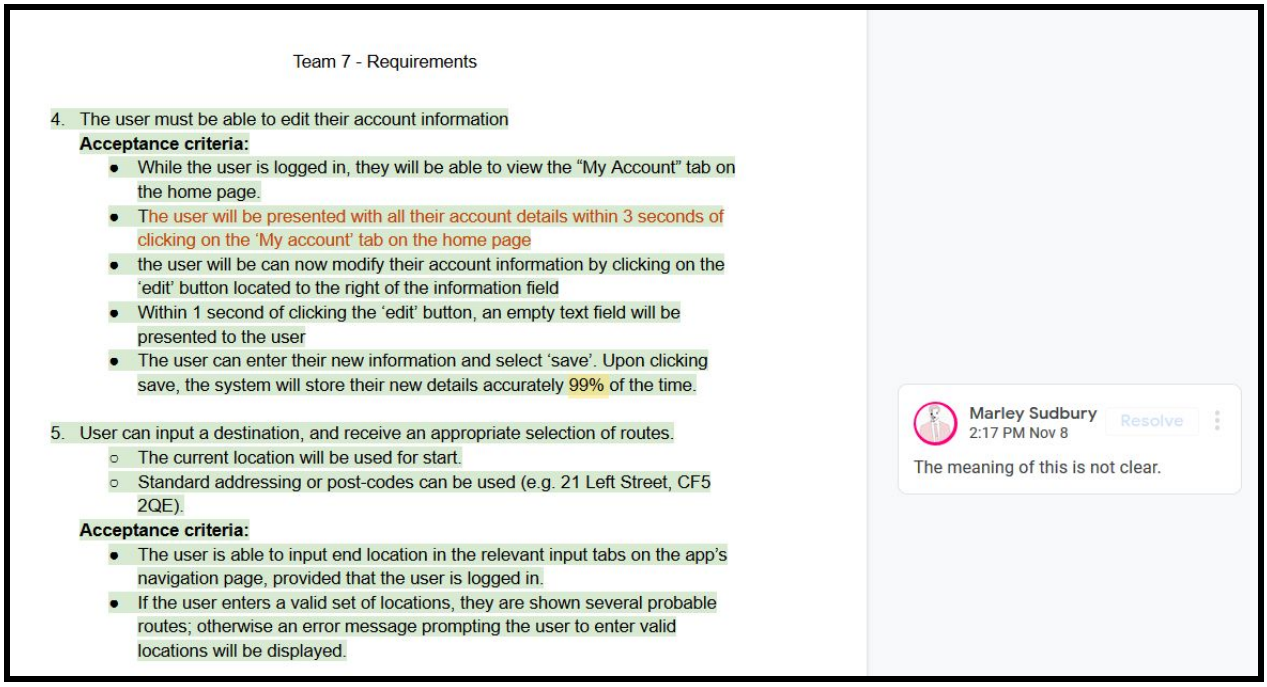
Subsequently, a group Google Drive folder was created, documenting our progress and allowing all members to access it.

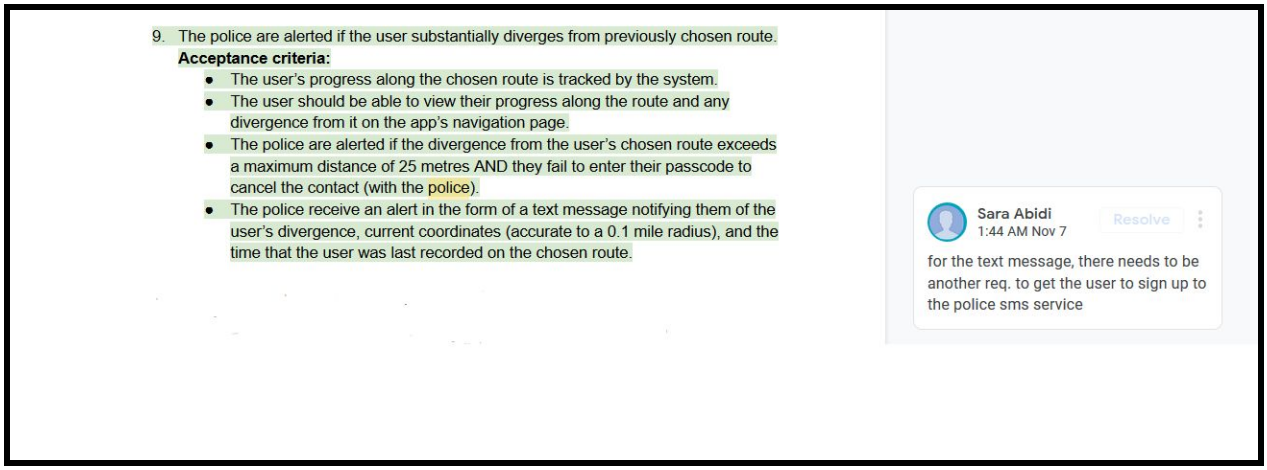


Our Google Drive folder as of 11/12/2019.

Requirements Presentation

For the requirements presentation we formed subteams, with each responsible for a specific section. Each pair worked collaboratively on their own Google Docs file. Whenever changes had to be made to somebody else's work, we would highlight the section of the document and add a comment; this would then be brought up and resolved during the next group meeting. We found this to be an excellent way to work together and avoid conflict within the group and hence, will continue this practice.





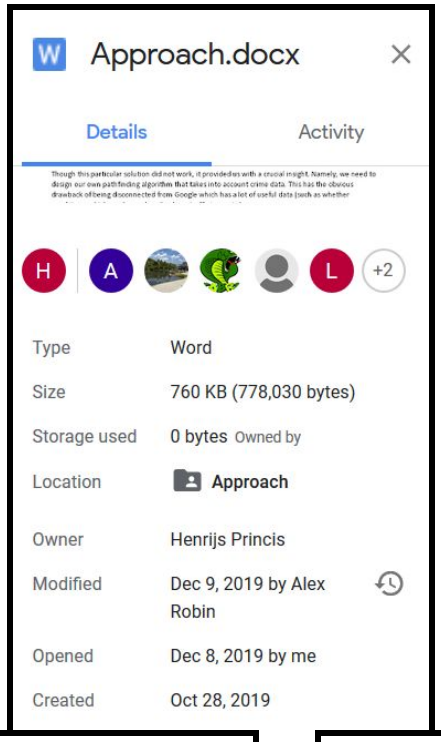
Examples of how we highlighted and wrote comments

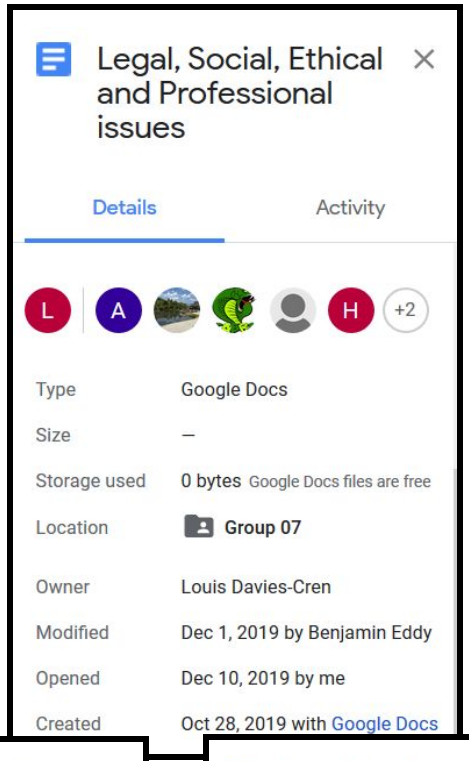
We had four pairs, each assigned one of the following:

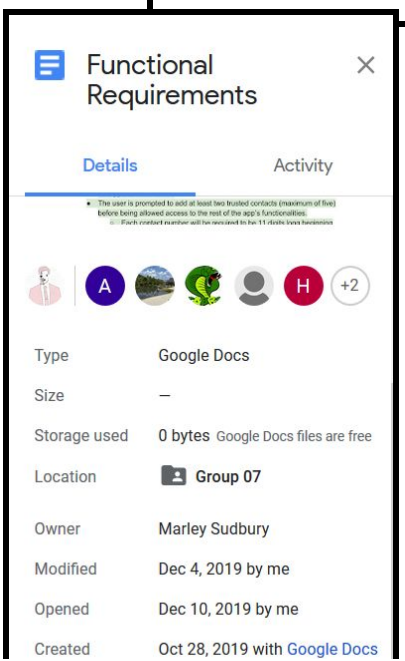
1. Approach to solving the problem
2. Requirements
3. Acceptance Criteria
4. Legal, Ethical and Social Issues

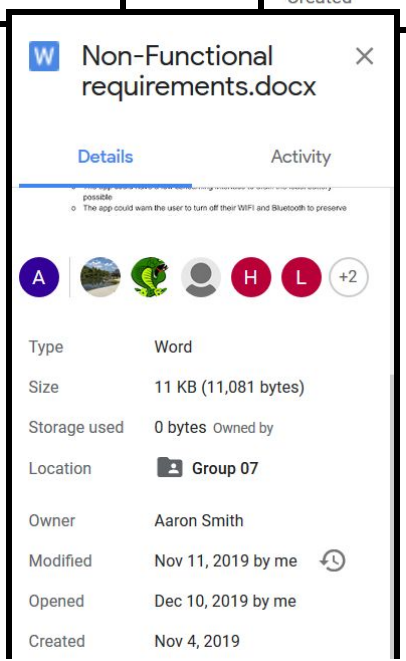
We spent the meetings leading up to the presentation working within our pairs; then, discussing any issues at the end with the group. We felt that resolving problems via group discussion was beneficial as we could consider each member's point of view and decide unanimously on how to move forward.

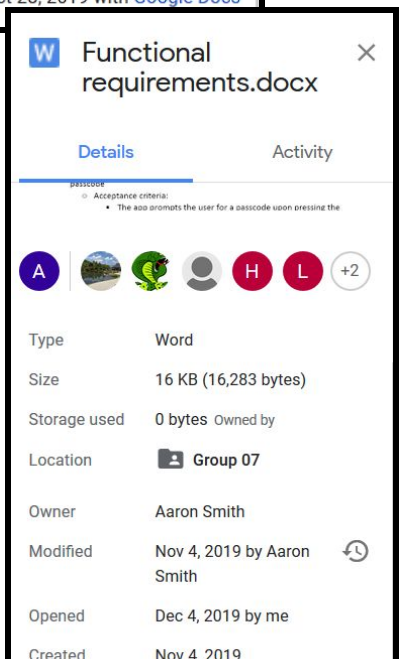
Once the sub-teams had finished their work, we had the following documents. Appendix B contains further screenshots.





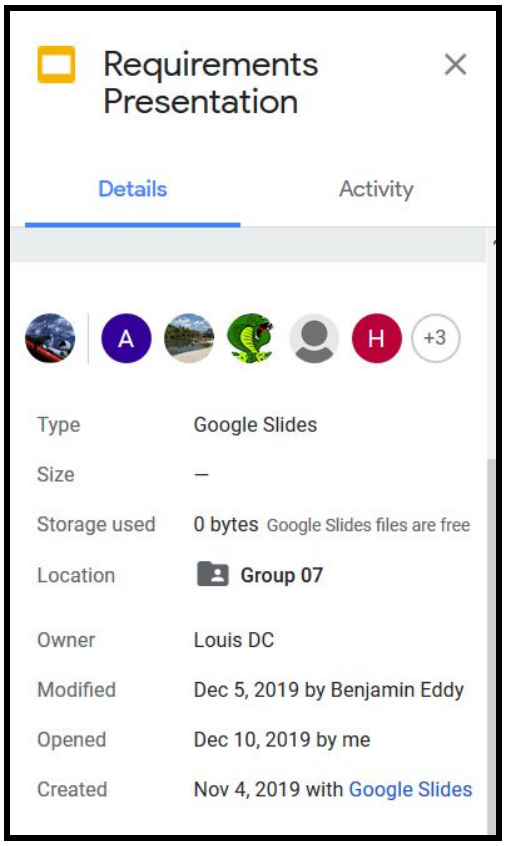






It became apparent that the requirements subteam and the acceptance criteria subteam's work overlapped. When allocating roles, we overlooked the fact that acceptance criteria is dependant upon functional requirements. The acceptance criteria subteam therefore devised their own list of requirements; consequently, both subteams worked on merging the two documents.

We presented each document during a group meeting where everybody could explain their progress, suggest changes and discuss next steps. We wanted to ensure that all group members were satisfied with each component before advancing.
To integrate each document into one presentation, we further polished our sections to make them fit for presenting.



We trimmed down the documents into a more digestible format for presenting, for example:

This explanation of the risk calculation

In the figure above, the path between the two edges would be evaluated to have a risk value of (100+200)/2 = 150. (Risk of taking the path (edge) between two nodes A and B would be modelled by the formula "RISK = (A+B)/2")

This simple approach is good because it is incredibly easy to calculate and gives a good estimate of the chances of being attacked, but it does not take into account the severity of the crime, the time of day it was committed (though our data currently does not contain this information, it is something we ideally we like to expand upon in the future.)

The severity of crime is easily solvable as the data points provided have the type of crime which means we can calculate the density of the crime for each node of different types of crimes:

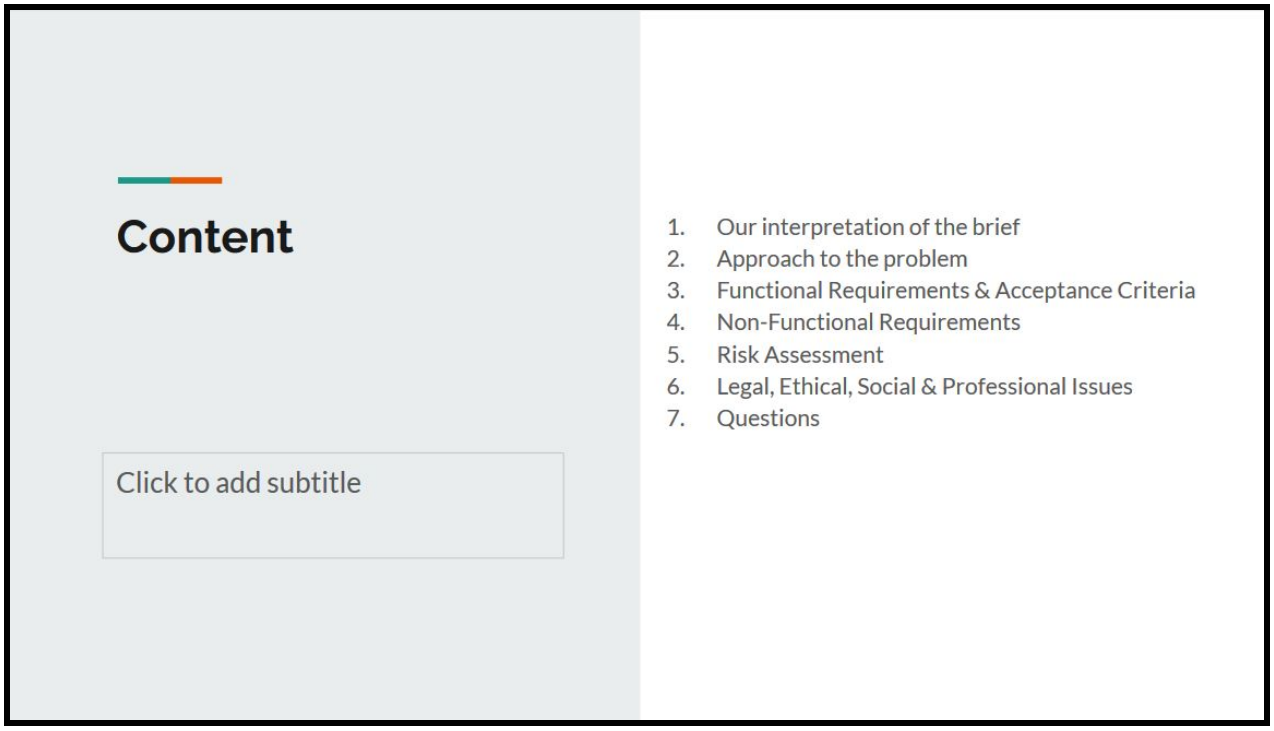
We can just divide total crime into subcategories and make a weighted risk value based on the weighted sum of these subcategories.

Became this slide of the presentation

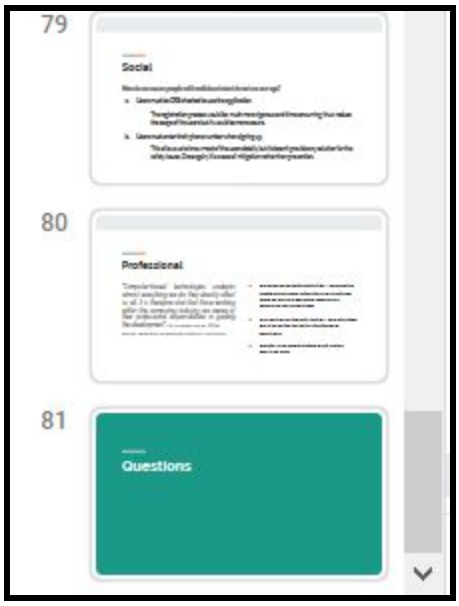
Safe navigation

- A length-risk bi-objective shortest path problem
- Apply A* or other path search algorithm to it
- Provide user with choices to minimise risk
- Deploy on online server

	Path 1	Path 2	Path 3	Path 4	Path 5
Length ℓ (m)	3955	4027	4060	4922	5988
Risk r ($\times 10^{-3}$)	2.32	2.02	2.01	1.71	1.70



New supplementary sections were added, such as the interpretation of the brief, and risk assessment. The final presentation contained 81 slides, with each subteam presenting their own sections.

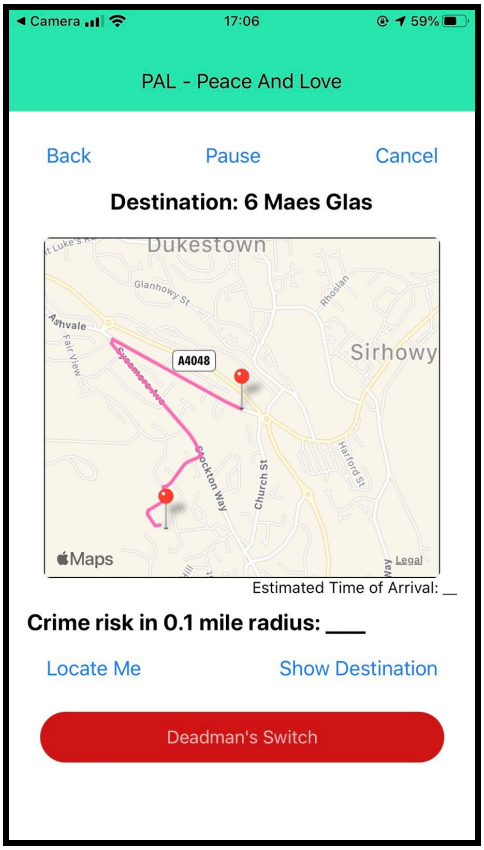


App Prototype

After the presentation, we focused on developing a prototype application, and working on the interim report. We chose to use React Native to develop our application. One key advantage of the framework is that it would allow us to implement the heat map prototype (see 'Evaluation Factors').

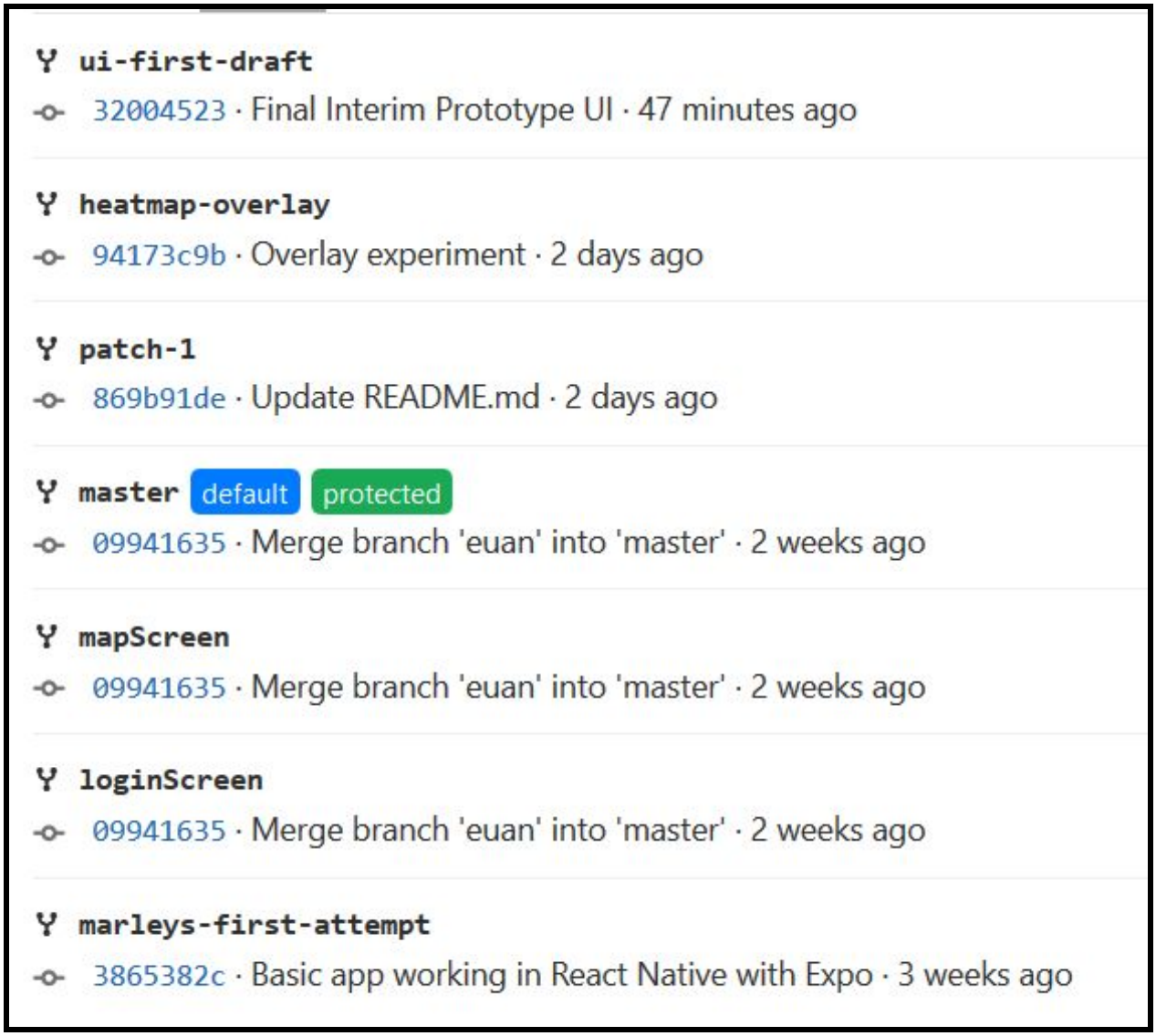
The UI was created to reflect the pre-existing design. (See “Detailed Design” and “Implementation - User Interface” sections of the report). The team collaborated on refining the design, and UI prototype.

The first version of the UI (on left) had no fullscreen button on the map. After some discussion, we decided this was too small to view and interact with effectively.



A GitLab repository was created for the prototype application. The master branch was set to protected, so each new section of code had to have its own branch. Only once completed should a merge request be made. Group members would discuss whether the changes before integrating it with the main code. Hence, one broken section wouldn’t cause problems in another.

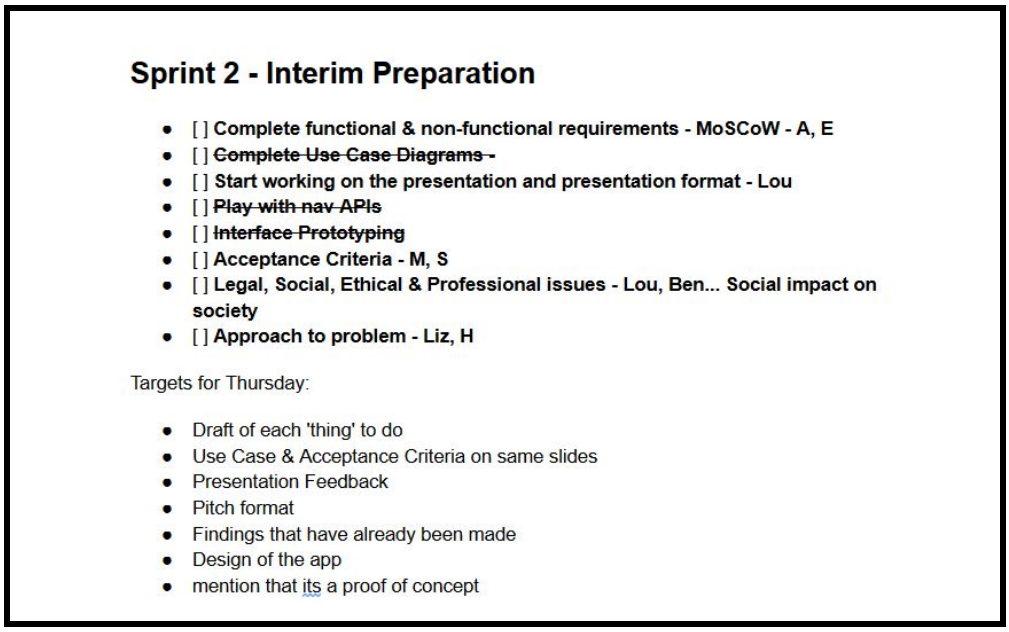
Our gitlab branches



Sprint Overview

We decided to use an Agile approach to the project, therefore we embarked upon a weekly sprint. A Sprint approach allowed us to focus on achieving a set amount of goals during a short period of time. We would use each Thursday meeting to review the previous Sprint, go over what was achieved and reflect upon the good and bad points. Finally, we would plan for the next sprint. We would write up a summary of each Sprint on our Google Drive.

Sprint 2 Screenshots

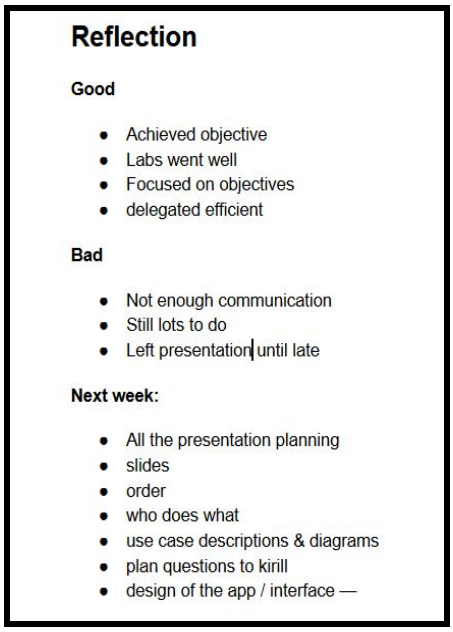


Sprint 2 - Interim Preparation

- [ ] **Complete functional & non-functional requirements - MoSCoW - A, E**
- [ ] ~~**Complete Use Case Diagrams -**~~
- [ ] **Start working on the presentation and presentation format - Lou**
- [ ] ~~**Play with nav APIs**~~
- [ ] ~~**Interface Prototyping**~~
- [ ] **Acceptance Criteria - M, S**
- [ ] **Legal, Social, Ethical & Professional issues - Lou, Ben... Social impact on society**
- [ ] **Approach to problem - Liz, H**

Targets for Thursday:

- Draft of each 'thing' to do
- Use Case & Acceptance Criteria on same slides
- Presentation Feedback
- Pitch format
- Findings that have already been made
- Design of the app
- mention that its a proof of concept



Reflection

Good

- Achieved objective
- Labs went well
- Focused on objectives
- delegated efficient

Bad

- Not enough communication
- Still lots to do
- Left presentation until late

Next week:

- All the presentation planning
- slides
- order
- who does what
- use case descriptions & diagrams
- plan questions to kirill
- design of the app / interface —

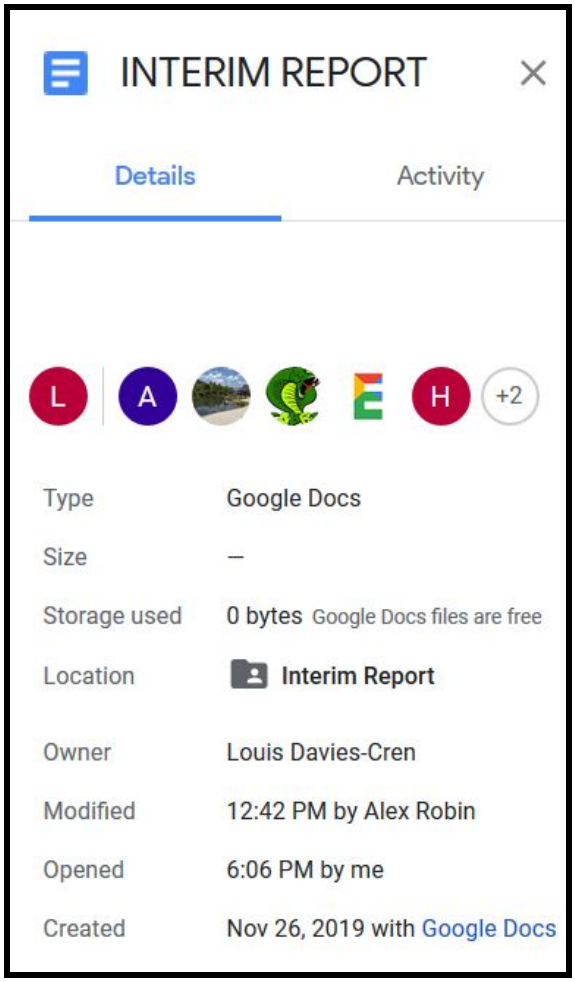
Summary

Below is a summary of all our sprints. For the complete write up of each sprint, see Appendix C.

Sprint #	Dates	Tasks
Sprint 1	26/10/19 to 01/11/19	1. Select the environment in which we are going to develop the software, including; programming languages, APIs, frameworks and Version Control 2. Draft requirements (functional and non-functional) 3. Assign roles 4. Create API tutorial
Sprint 2	02/11/19 to 08/11/19	1. Complete functional and non-functional requirements 2. Start working on presentation 3. Acceptance criteria 4. Legal, Social Ethical & Professional issues 5. Explanation of approach to the problem
Sprint 3	09/11/19 to 15/11/19	1. Review presentation 2. Polish presentation 3. Rehearse presentation
Sprint 4	16/11/19 to 22/11/19	1. Reflecting on feedback on presentation 2. Further discussion of frameworks to be used 3. Setting up the development environment on all team members' devices
Sprint 5	23/11/19 to 29/11/19	1. Delegating roles for Interim Report & parallel prototyping 2. Establishing guidelines for executing report. 3. Discussion of time plan for interim report
Sprint 6	30/11/19 to 08/12/19	1. Working towards completion of the report 2. Continuous feedback on report content

Interim Report

We began writing the Interim Report using the same principles I've previously discussed since they helped us work efficiently. We split up the project based on the criteria for assessment, and tried to distribute the workload evenly. Working collectively on one main Google Docs file for the report, we planned to finish each of our sections by Monday December 9th, leaving our final three group meetings to proofread our report as a group.


INTERIM REPORT
Details
Activity
L
A
H
+2
Type
Google Docs
Size
—
Storage used
0 bytes
Google Docs files are free
Location
Interim Report
Owner
Louis Davies-Cren
Modified
12:42 PM by Alex Robin
Opened
6:06 PM by me
Created
Nov 26, 2019 with Google Docs

References

Alibaba. 2019. *Java Coding Guidelines*. Available at: https://alibaba.github.io/Alibaba-Java-Coding-Guidelines/ [Accessed: 11 December 2019].

Anderson, T.K. 2009. Kernel density estimation and K-means clustering to profile road accident hotspots. *Accident Analysis & Prevention* 41(3), pp. 359–364. doi: 10.1016/j.aap.2008.12.014.

Apple. 2019. *Terms and Conditions*. Available at: https://developer.apple.com/terms/ [Accessed: 11 December 2019].

Ben Ticha, H. et al. 2017. A solution method for the Multi-destination Bi-objectives Shortest Path Problem. doi: 10.13140/RG.2.2.32492.05768.

Conway, M.E. [No date]. *Conway's Law*. Available at: http://www.melconway.com/Home/Conways_Law.html [Accessed: 12 December 2019].

Draw.io. 2018. *UML Diagrams*. Available at: https://about.draw.io/uml-diagrams/ [Accessed: 12 December 2019].

European Parliament and the Council of the European Union. 2016. *Right to be Forgotten*. Available at: https://gdpr-info.eu/issues/right-to-be-forgotten/ [Accessed: 10 December 2019].

European Parliament and the Council of the European Union. 2016. *Right to be Informed*. Available at: https://gdpr-info.eu/issues/right-to-be-informed/ [Accessed: 10 December 2019].

Facebook. [No date]. *Testing React Native Apps*. Available at: https://jestjs.io/docs/en/tutorial-react-native/ [Accessed: 12 December 2019].

Galbrun, E. et al. 2014. Safe Navigation in Urban Environments. *3rd International Workshop on Urban Computing at KDD 2014, UrbComp'14*

Google Play. 2018. *User Data*. Available at: https://play.google.com/about/privacy-security-deception/ [Accessed: 10 December 2019].

Google. 2019. *Java Style Guide*. Available at: https://google.github.io/styleguide/javaguide.html [Accessed: 11 December 2019].

Mallawaarachchi V. 2017. *10 Common Software Architectural Patterns in a Nutshell*. Available at: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013/ [Accessed: 12 December 2019].

Matanov, I. 2018. *Top 10 Things to Consider When Developing a Mobile App*. Available at: https://expertise.jetruby.com/top-10-things-to-consider-when-developing-a-mobile-app-97ab00e664bf/ [Accessed: 10 December].

Meier, J. D. 2008. *Layers and Components*. Available at: https://blogs.msdn.microsoft.com/jmeier/2008/09/06/layers-and-components/ [Accessed: 12 December 2019].

NSW Attorney General's Department. 2011. *Displacement theory*. Available at: http://www.crimeprevention.nsw.gov.au/Documents/displacement_theory_factsheet_oct2014.pdf [Accessed: 10 December 2019].

Office for National Statistics. 2017. *Crime in England and Wales: year ending March 2017*. Available at: https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/bulletins/crimeinenglandandwales/yearendingmar2017 [Accessed: 10 December 2019]

Phillips, H. 2018. Risk Management. *Developing Quality Software*. Available at: https://learningcentral.cf.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_387499_1&content_id=_4873211_1 [Accessed: 10 December 2019].

Police UK. 2019. Detailed Statistics for Cardiff. Available at: https://www.police.uk/south-wales/swpcardiff/crime/stats/ [Accessed: 8 December 2019].

React Native Community. 2019. *React-Native-Maps*. Available at: https://github.com/react-native-community/react-native-maps/ [Accessed: 8 December 2019].

React Native Docs, 2019. *Native Modules* at: https://facebook.github.io/react-native/docs/native-modules-android [Accessed: 8 December 2019].

Richards, M. 2015. *Software Architecture Patterns* [eBook version]. [No place]: O'Reilly. Available at: https://learning.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html [Accessed: 12 December 2019].

Sedeño-Noda, A. and Raith, A. 2015. A Dijkstra-like method computing all extreme supported non-dominated solutions of the biobjective shortest path problem. Computers & Operations Research 57, pp. 83–94. doi: 10.1016/j.cor.2014.11.010.

Smartdraw. [No date]. *Use Case Diagram*. Available at: https://www.smartdraw.com/use-case-diagram/ [Accessed: 12 December 2019].

Tallyfy. [No date]. *All You Need to Know About UML Diagrams: Types and 5+ Examples*. Available at: https://tallyfy.com/uml-diagram/ [Accessed: 12 December 2019].

Voip-info.org. [No date]. *CNAM*. Available at: https://www.voip-info.org/cnam/ [Accessed: 10 December 2019].

Wu, James 2019. *How scrum help turn around our development process*. Available at: https://medium.com/@jw207427/how-scrum-help-turn-around-our-development-process-dac6ff7c700 [Accessed: 12 December 2019].

Appendices

Appendix A—Test Cases

Test Case ID: 1			
Test Purpose: The user can create an account.			
Environment: Expo client app (>=2.14.0) running on Android (>=10.0).			
Preconditions: The user has the app installed but is not logged in and has no existing account associated with their phone number.			
Test Case Steps:			
Step No.	**Procedure**	**Expected Results**	**P/F**
1	On the login screen, press the button which says ‘Create Account’.	A form will appear for the user to fill in.	
2.1	Input the phone number of the test device in the Phone Number input.	A tick mark (e.g. ✓) will signify that the phone number is valid.	
2.2	Input the password ‘TestingPassword52563’ into the Password and Verify Password inputs.	The passwords will appear as bold circles (e.g. •), and a tick mark (e.g. ✓) will signify that the first input meets requirements and that they are the same.	
2.3	Enter ‘16/1/1990’ in the Birth Date input.	A tick mark (e.g. ✓) indicates that the birthdate is valid (i.e. the user is 18+).	
2.4	Press the Create Account button.	A form appears asking for a number that the user will receive via SMS.	
3	Enter the verification code once received via SMS.	The main menu appears.	
Comments			
Author: Marley	**Date:** 06/12/19	**Checker:**	**Date:**

Test Case ID: 2			
Test Purpose: The user can login with a previously existing account.			
Environment: Expo client app (>=2.14.0) running on Android (>=10.0).			
Preconditions: An account exists with the credentials ‘07700 900007’ and ‘TestingPassword52563’.			
Test Case Steps:			
Step No.	**Procedure**	**Expected Results**	**P/F**
1.1	On the login screen enter ‘07700 900007’ in the Phone Number input.	The phone number entered should be shown.	
1.2	Enter ‘TestingPassword52563’ in the password input.	The password will appear as bold circles (e.g. •).	
2	Press the Login button.	The main menu appears.	
Comments			
Author: Marley	**Date:** 06/12/19	**Checker:**	**Date:**

Test Case ID: 3			
Test Purpose: The user can successfully start a journey.			
Environment: Expo client app (>=2.14.0) running on Android (>=10.0).			
Preconditions: The user is logged in and has at least two trusted contacts.			
Test Case Steps:			
Step No.	**Procedure**	**Expected Results**	**P/F**
1	From the main menu select ‘Navigation’.	The navigation screen will appear.	
2	Press ‘Select Destination’.	A text field will appear for the user to input a destination.	
3	Enter ‘Cardiff University Students’ Union, Park Place, Cardiff, CF10 3QN’.	A list of possible routes from the user’s current location to the Student’s Union will appear on the screen.	
4	Select the first route in the list.	The journey will begin, with a standard SatNav style display guiding the user.	
Comments			
Author: Marley	**Date:** 06/12/19	**Checker:**	**Date:**

Test Case ID: 4			
Test Purpose: The user can finish a journey successfully.			
Environment: Expo client app (>=2.14.0) running on Android (>=10.0).			
Preconditions: The user has already started a journey (see Test 3).			
Test Case Steps:			
Step No.	**Procedure**	**Expected Results**	**P/F**
1	Follow the directions presented on the screen at walking speed until you reach the destination.`	The map in the app moves with the user and then displays ‘Journey Complete’.	
Comments			
Author: Marley	**Date:** 06/12/19	**Checker:**	**Date:**

Test Case ID: 5			
Test Purpose: Alarm will be raised if the user significantly diverges from chosen route.			
Environment: Expo client app (>=2.14.0) running on Android (>=10.0).			
Preconditions: The user has already started a journey (see Test 3).			
Test Case Steps:			
Step No.	**Procedure**	**Expected Results**	**P/F**
1	Instead of following the directions provided by the app, walk 25 metres in the opposite direction.	Once the GPS registers that the device is more than 25 metres away from the selected route it will tell the user to return to the path.	
2	Ignore the message and remain off of the path. Wait for the countdown to begin.	After a length of time determined by the users settings the app will start an alert countdown and ask the user to input their password to disable the alert.	
3	Ignore the message and wait for the countdown to end.	After the countdown ends, a message is sent to the emergency contacts or police depending on the users settings.	
Comments			
Author: Marley	**Date:** 08/12/19	**Checker:**	**Date:**

Test Case ID: 6			
Test Purpose: The user can pause and resume a journey that they are on.			
Environment: Expo client app (>=2.14.0) running on Android (>=10.0).			
Preconditions: The user has already started a journey (see Test 3).			
Test Case Steps:			
Step No.	**Procedure**	**Expected Results**	**P/F**
1	Press the pause button at the top of the page.	A prompt will appear asking for the user password.	
2	Enter the password associated with the account.	Text at the top of the screen will indicate to the user that their journey has been paused. The map and route will still be on the screen, but will not respond to the user's GPS. The button at the top of the screen which previously said 'Pause' will now say 'Resume'.	
3	Press the resume button.	The previous journey will be resumed if the user is still on the route, otherwise they will be prompted to select a new route to the destination.	
Comments			
Author: Marley	**Date:** 09/12/19	**Checker:**	**Date:**

Test Case ID: 7			
Test Purpose: The user can cancel a journey that they are on.			
Environment: Expo client app (>=2.14.0) running on Android (>=10.0).			
Preconditions: The user has already started a journey (see Test 3).			
Test Case Steps:			
Step No.	**Procedure**	**Expected Results**	**P/F**
1	Press the cancel button at the top of the screen.	A prompt will appear asking for the user's password.	
2	Enter the password associated with the account	A pop-up will tell the user they have successfully cancelled their journey.	
Comments			
Author: Marley	**Date:** 10/12/19	**Checker:**	**Date:**

Appendix B—Further Evidence

Pre-presentation documentation (first pages only)

Legal, Social, Ethical and Professional issues

Main issues are identified and clearly related to the specific problem

Legal

Need to abide by DPA, GDPR, Google Play Store, Apple App store

- GDPR's "Right to be Forgotten": Keeping in line with article 17, once it is confirmed that a user's journey has been completed safely his/her GPS data from that journey will automatically be deleted. If a users data isn't needed it'll have to be removed
- The only case where the user's location will be stored & distributed beyond the central server is:
    a. If their device has low battery and they have enabled to notify a friend of their location
    b. If the device senses that an attack has occurred
    c. Maybe: if we feed safe routes back into the system

- GDPR's "Right to be Informed": We must update users when there has been an update on our privacy policy and the data that we'll be gathering. Must keep in mind for future updates.
- How their GPS will be used if we were to track safe & unsafe journeys
- Need to clarify that we have no responsibility if they get attacked - very clear
- https://pdfs.semanticscholar.org/7f9f/71f8fa74c83f6e8aa130e7d217e0531c1708.pdf

Google play

- **Cannot** only be placed in a privacy policy or terms of service; and
- **Cannot** be included with other disclosures unrelated to personal or sensitive data collection

Means we have to have a pop up explain to the user how we are going to use their data.#

Approach

Problem interpretation

We are required to make an app that minimizes the chances of being attacked at night when travelling and mitigates the impact of an attack.

Minimizing chances of attack

The aspects we gave weight to the most were

- Minimizing the time spent when travelling at night (shorter paths are preferred)
- Considering the chances of being attacked when taking a certain path (some streets tend to have more crime, but also population needs to be considered)
- Different crimes have different severity levels (violent crime is generally worse than a misdemeanor and therefore should not count as strongly)
- The time of day the crime was committed
- Giving the user the ability to choose between risk and time spent travelling
- Encouraging safety in groups (if multiple people are taking the same path, we should group them)

Mitigating the impact of an attack

- Reliably detecting when an attack has occured
- Detecting an attack without the user's input
- Making the app quickly respond to when the user is being attacked
- Giving useful information like location to the people responsible for your safety (police/friends)

Reasoning behind minimizing chances of attack

We first decided to implement a system for minimizing the chances of being attacked as we felt that it was the main part of our brief and we did not want to be limited by the restrictions placed on implementing the mitigation part. Therefore, we had free choice over the programming languages and APIs we could use to achieve the task at hand. We considered the simpler approaches first:

Functional Requirements

MUST

1. The user must be able to make an account.
   Acceptance criteria:
   - Upon opening the app, the user is presented with the login page within 5 seconds, which will also include a 'Create an account' button within.
   - On selecting the button, the user will be prompted for their name, phone number, age and passcode in the provided input tabs.
     - The passcode will have to be verified by entering the field twice.
   - The user must be 18 because of grouping.
   - The user will receive a verification pin via text which they will have to enter into the provided input tab to verify their account.

2. The user must be able to log in to the app.
   Acceptance criteria:
   - Within 5 seconds of the user launching the application, they will be prompted for their phone number and password.
   - If they enter correct information, the user will then be granted access to the apps functionality.
   - If the information is incorrect, the user will be presented with a 'Sorry! Your phone number or passcode is incorrect' and will be prompted within 2 seconds for their details again.

3. The user must add at least two trusted contacts before using the app.
   Acceptance criteria:
   - An option to add trusted contacts is presented to the user once they sign in to the app for the first time.
   - The user is prompted to add at least two trusted contacts (maximum of five) before being allowed access to the rest of the app's functionalities.
     - Each contact number will be required to be 11 digits long beginning with '07'

Non-functional

- The systems database must be secure from people outside of the development team as sensitive GPS data and personal data is being stored.
- The system should load the location route within 10 seconds of the user's request on a device post 2016.
- The interface should be simple and easy to navigate
    - Easy to read font
    - Clearly labelled buttons
    - Most users should be able to understand how to use the app within 5 minutes of their first use.
        - The app should contain help sections to provide the user with detailed usage information
    - All menus must have a consistent layout
- maintainable
    - Ability to change code if the client requests.
    - The source code should be backed up on one physical device and a cloud service.
- Should run smoothly on all mobile devices (For prototype, mac, Linux and windows).
    - The app start-up cycle should last no more than 10 seconds on a smartphone post 2016 model. (IOS / Windows / Linux version for prototype)
- Users data must be stored securely
    - The login details could be encrypted by the system
    - All user data should be backed up weekly
- The system should update its crime data on the first day of every month to ensure relevance.
- User accounts must have strong passwords to be accepted, this means 8 or more characters including upper/lower case letters, numbers and symbols.
- All user input must be parsed correctly to ensure there are no unhandled exceptions

Appendix C—Sprints

Sprint 1 - Planning

- [ ] **Select a language and modules**
- [ ] **Refine requirements (functional & non-functional)**

    1. User can see the safest route to the destination they put in
    2. User can select the route depending on how safe or time consuming it is (ideally 3 options)
    3. Can enable a setting that when they deviate from the path, it alarms a friend or contact
    4. Can enable a setting where sensors (gyro, accel or touch screen) can activate a warning or alarm1
- Use MOSCOW - Must, Should, Could, Would/Won't not sure

  Clarification of an acceptance criteria for the use case

- [ ] Assign roles
  Scrum Master - Louis Davies-Cren
- [x] API tutorial
- [x] Set up github env

Reflection

Good:

- better idea of how it will work
- progress above average
- Heatmap

Bad:

- Have not fulfilled the goals (Use Case)
- More communication on when to upload docs etc
- Haven't been able to organise our client meeting
- Make GitHub repo private

Sprint 2 - Interim Preparation

- [ ] **Complete functional & non-functional requirements - MoSCoW - Aaron, Euan**
- [ ] ~~**Complete Use Case Diagrams -**~~
- [ ] **Start working on the presentation and presentation format - Louis**
- [ ] ~~**Play with nav APIs**~~
- [ ] ~~**Interface Prototyping**~~
- [ ] **Acceptance Criteria - Marley, Sara**
- [ ] **Legal, Social, Ethical & Professional issues - Louis, Ben... Social impact on society**
- [ ] **Approach to problem - Lizheng, Henrijis**

Targets for Thursday:

- Draft of each 'thing' to do
- Use Case & Acceptance Criteria on same slides
- Presentation Feedback
- Pitch format
- Findings that have already been made
- Design of the app
- mention that it's a proof of concept

Reflection

Good

- Achieved objective
- Labs went well
- Focused on objectives
- delegated efficient

Bad

- Not enough communication
- Still lots to do
- Left presentation until late

Next week:

- All the presentation planning
- slides
- order
- who does what
- use case descriptions & diagrams
- plan questions to kirill
- design of the app / interface —

Sprint 3

Presentation format

1. Clarification of how we've interpretation the brief/problem
2. Approach to the problem
3. Functional & Non-Functional Requirements
4. Acceptance Criteria
5. Legal, Ethical, Social & Professional Issues with approach
6. Questions

Reflection

Good

- Presentation content is at a high standard
- managed to do multiple run-throughs
- Despite disagreement, the group came back together, understood the content more & knows how to handle conflict well
- first try at the presentation was a good start

Bad

- Our first disagreement
- Presentation wasn't produced by deadline
- Haven't used kanban well enough
- make sure speaker notes are available

TO-DO:

Presentation

Submit presentation - 18th of November

Sprint 4

- JAVA - need to pick a framework {PROOF OF CONCEPT}
- OpenStreetMap
- Drag & Drop ?
- Delegating roles / make sub-teams ?

Interim Reports

40% - Overall 80% for group 20% for individual

15% - Requirements Presentation

45% - End Report

To do:

- Make REPO
- Make Fake Map
- Create Menus
- Log-In/Sign Up
- Dead Man's switch
- Help Button & Contact Page

Reflection

Good

- Very good presentation
- Received good feedback
- 'Started' developing
- Most of us have installed the software

Bad

- Arbitrarily chose our platform, we're quite naive at the moment.
- Forgot to book rooms
- Meetings have been less productive
- Yet to start the report

Sprint 5 and Sprint 6

1 main coding team

1 main report team

System Requirements - reformatted (aaron)

- Non-functional & functional

Software Architecture and data design - needs to be done (Sara)

- Diagrams etc

Detailed design - needs to be done (Aaron)

- Prototyping, data flow

Implementation - (Liz)

- Clear Readable code, consistent style & commenting
- CamelCase
- Pseudocode / prototyping

Testing - (Marley)

- Test cases, plan them based on requirements, acceptance criteria & prototypes

Evidence that work is implemented - (Euan)

- Screenshots, gitlab links

Justification & Evaluation 40%

- Like beginning of the presentation (Henry)
- Strengths & Limitations (Louis)
- Legal, Social, Ethical & Professional issues (Ben)

Work Plan 10% - LOUIS

Report Writing 10% - MARLEY

- Formatting it once everyone has contributed

Rules:

1. Don't remove other people's work. Instead, add your own version underneath and bring it up at the next meeting or message the gc / individual
2. Follow the headings that have been put in place by putting the content in the right sections
3. Avoid working on a local version / update the shared version as often as possible
4. Use gender neutral terms when referring to actors (users, developers etc)
5. References: Keep within the appropriate section for now (until finalised)

Good

- Started the report
- Allocated tasks
- Set ground rules for docs

Bad

- Not much on the report
- Deadline in 2 weeks
- Marley's late
- Ben not here

P.A.L

Sprint 7 - REPORT WRITING

Good

- Made good progress on the report
- UI is looking good
- Marley made the template for the test cases
- Received presentation marks ! 10/10

Bad

- People beginning to feel burnt out (not due to GP stuff though)
- Marley is leaving uni early so we need to prioritise his part of the report first

Appendix D—Code Style Guideline

A work majorly based on Google Java Style Guide and Alibaba Java Coding Guideline, which is distributed under CC-By 3.0 License and Apache 2.0 License respectively.

1. Characters

1.1. The ASCII horizontal space character (0x20) is the only whitespace character that appears anywhere in a source file. No tab should be used for indentation.

2. Naming

2.1. A package should be named in lowercase characters. There should be only one English word after each dot. Package names are always in singular format while class names can be in plural format if necessary.

example:

`com.google.open.util` can be used as a package name for utils; `MessageUtils` can be used as a class name..

2.2. Uncommon abbreviations should be avoided for the sake of legibility.

Counter example:

AbsClass (AbstractClass); condi (Condition)

2.3. The pattern name is recommended to be included in the class name if any design pattern is used.

Positive example:

`public class OrderFactory;`

`public class LoginProxy;`

`public class ResourceObserver;`

2.4. Class names are written in UpperCamelCase.

2.5. Method names, parameter names, member variable names, and local variable names should be written in lowerCamelCase.

Positive example:

localValue / getHttpMessage() / inputUserId

2.6. Constant variable names should be written in upper characters separated by underscores. These names should be semantically complete and clear.

Positive example:

MAX_STOCK_COUNT

Counter example:

MAX_COUNT

3. Format

3.1. For Java code, an indent is 4 spaces long.

3.2. For Nonempty blocks, using K & R style:

- No line break before the opening brace.
- Line break after the opening brace.
- Line break before the closing brace.
- Line break after the closing brace, only if that brace terminates a statement or terminates the body of a method, constructor, or named class. For example, there is no line break after the brace if it is followed by else or a comma.

4. OOP Rules

4.1. An overridden method from an interface or abstract class must be marked with `@Override` annotation.

4.2. Limit the accessibility of class,method and field as much as possible. All fields should be wrapped by getter and setter. The setter is only added if it is mutable.

4.3. If a class is a util class, initialization should be banned by throwing an exception.

5. Comments

5.1. All methods should have comments to describe their input, output, and what it does as a black box. Describe implementation details only if necessary.

5.2. Tags in comments (e.g. TODO, FIXME) need to be handled and cleared in time by scanning frequently.

5.2.1. TODO means the logic needs to be done, but not finished yet. Actually, TODO is a member of Javadoc, although it is not realized in Javadoc yet, but has already been widely used. TODO can only be used in class, interface and methods, since it is a Javadoc tag.

5.2.2. FIXME is used to represent that the code logic is not correct or does not work, should be fixed in time.

5.3. Javadoc should be used for classes, class variables and methods. The format should be '/** comment \n @annotation comment **/', rather than '// xxx'.

5.4. Every class should include information of its author(s).

5.5. Single line comments in a method should be put above the code to be commented, by using // and multiple lines by using /* */. Alignment for comments should be noticed carefully.

6. Logs

6.1. Use Logging at highly risky points, such as exception handling and network communication.

6.2. Do not use API in log system (Log4j, Logback) directly. API in log framework SLF4J is recommended to use instead, which uses Facade pattern and is conducive to keep log processing consistent.