

# Population of Environment Evaluation and Prediction System

---

An investigation into crowd-sourced population tracking



Presented by:  
**Ian Grant**

Prepared for:  
**Dr. S. Winberg**  
Dept. of Electrical and Electronics Engineering  
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town  
in partial fulfilment of the academic requirements for a Bachelor of Science degree in  
Electrical and Computer Engineering.

**November 11, 2020**



## Declaration

---

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

A handwritten signature in black ink, appearing to read 'Ian P. S. Grant', is written over a light gray rectangular grid background.

Signature:.....

Ian P. S. Grant

Date:        November 11, 2020

# Acknowledgments

---

Any project of this scale could not be completed without the contributions of many individuals. I would like to use this space to acknowledge those who sacrificed much for me to be able to pursue my passion for electronics and who assisted in the formation of this project.

Firstly, I would like to thank my project supervisor, Dr. Winberg, for being a supportive mentor in this endeavour. His enthusiasm and expertise have been a resource that has been invaluable to the creation of PEEPS.

Secondly, I would like to thank my peers in the Electrical Engineering Department for fostering my passion for engineering, and for the support I have always been given whenever I have needed it.

Thirdly, I would like to thank the University of Cape Town and the Electrical Engineering Department faculty for providing the resources I have needed to pursue a life surrounded by innovation, science, and creation. Their tutelage and support over the last four years will not be forgotten.

Lastly, but most importantly, I would like to thank my family. Without their dedication, love and sacrifice, I would never have been able to achieve what I have been able to. I am privileged to be in environment where I am encouraged to take risks and explore my interests, while also being supported regardless of whether I succeed or fail. I would like to thank James and Emily in particular for keeping my spirits high no matter the circumstances.

# Abstract

---

Recognising the pressing need for an exploration of population density prediction mechanisms brought to light by COVID-19, this project aims to investigate the design and implementation of crowd-sourced population evaluation mechanisms. This project uses a comprehensive review of the app development, location tracking, and data acquisition literature in order to develop a robust population tracking smartphone application framework, named the Population of Environment Evaluation and Prediction System (PEEPS). This application implements user recognition, location sharing, and data processing to provide users with actionable information about the density of localised populations. Additionally, the smartphone app provides the user with insight into how many people they are likely to encounter throughout the course of the day. PEEPS is used as a mechanism to investigate the development of a crowd-sourced and location-predictive application. Additionally, the application undergoes both functional and investigative testing to determine its abilities and shortcomings. This project intends to inform developers who wish to implement crowd-sourced location prediction about the design process and implementation of such an application, while also providing such developers with the necessary insights and codebase to jumpstart development. The testing that was conducted on the application proved the app's ability to provide potential users with information that is accurate, timely and actionable. In addition, the testing determined that the app used insubstantial smartphone resources, thus proving its ability to act as a scalable framework for developers to utilise in their own location prediction applications. Lastly, the project provides recommendations gathered from the investigative testing and presents opportunities for future development.

# Contents

Glossary . . . . .	viii
<b>1 Introduction</b>	<b>1</b>
1.1 Background to the Study . . . . .	1
1.2 Objectives of this Study . . . . .	2
1.3 Scope and Limitations . . . . .	2
1.4 Plan of Development . . . . .	3
<b>2 Smartphones as Population Density Trackers</b>	<b>4</b>
2.1 Smartphone Access and Development . . . . .	4
2.1.1 App Development . . . . .	5
2.1.2 Smartphone Technology . . . . .	5
2.2 Location Tracking . . . . .	6
2.2.1 Methods of Tracking . . . . .	7
2.2.2 Existing Services . . . . .	7
2.2.3 Location Tracking APIs . . . . .	8
2.2.4 Accuracy and Reliability . . . . .	9
2.3 Data Acquisition . . . . .	10
2.3.1 Crowd Sourced Data . . . . .	10
2.3.2 Data Storage and Server Solutions . . . . .	11
2.3.3 Smartphone User Privacy . . . . .	12
<b>3 PEEPS Development Methodology</b>	<b>14</b>
3.1 Problem Definition . . . . .	14
3.2 Development Strategy . . . . .	15
3.3 Project Requirements . . . . .	16
3.4 Project Constraints . . . . .	17
3.5 Technical Specifications . . . . .	17
<b>4 Design and Implementation</b>	<b>20</b>
4.1 Conceptual Design . . . . .	20
4.1.1 Concept Design Proposals . . . . .	20

4.1.2	Final Design Decision . . . . .	24
4.2	The PEEPS Application Structure . . . . .	25
4.2.1	System Level Overview . . . . .	25
4.2.2	Inter-System Interaction Summary . . . . .	26
4.2.3	Data Anonymity . . . . .	26
4.3	App Design and Development . . . . .	26
4.3.1	System Description . . . . .	26
4.3.2	Java Class Structure . . . . .	27
4.3.3	Application Services . . . . .	29
4.3.4	Subsystem Analysis . . . . .	31
4.3.5	Population Prediction Functionality . . . . .	33
4.4	Server Design and Development . . . . .	37
4.4.1	System Description . . . . .	37
4.4.2	Subsystem Analysis . . . . .	37
4.4.3	Web Interface Structure . . . . .	38
4.4.4	Database Architecture . . . . .	41
<b>5</b>	<b>Testing and Results</b>	<b>42</b>
5.1	Final Product Presentation . . . . .	42
5.1.1	Application Interaction Feedback Mechanisms . . . . .	44
5.2	Functional Testing . . . . .	45
5.3	Investigative Testing . . . . .	46
5.4	Verification of Specifications . . . . .	49
5.5	Discussion . . . . .	51
<b>6</b>	<b>Conclusions</b>	<b>52</b>
<b>7</b>	<b>Recommendations and Future Development</b>	<b>54</b>
7.1	Recommendations . . . . .	54
7.1.1	Flutter as a Substitute for Java . . . . .	54
7.1.2	Fullscreen Dialog-Fragment . . . . .	54
7.1.3	Data Processing . . . . .	55
7.2	Future Development . . . . .	55
7.2.1	Extensive Network Testing . . . . .	55
7.2.2	Population-Scale Testing . . . . .	55
7.2.3	iOS Comparison Testing . . . . .	56
<b>A</b>	<b>Additional Files</b>	<b>60</b>
A.1	TEST-2 Data and Control . . . . .	60

<b>B</b>	<b>Addenda</b>	<b>61</b>
B.1	Github link . . . . .	61
B.2	Ethics Form . . . . .	61



# List of Figures

3.1	Project phase breakdown using the Waterfall Model. . . . .	15
4.1	Android Studio’s activity templates used to jumpstart development. . . .	21
4.2	High level overview of the main application systems and their components.	25
4.3	Java class hierarchy for the PEEPS App. . . . .	27
4.4	Location Fragment displaying a user’s saved locations with their popula- tion status. . . . .	34
4.5	Add-Location View with ‘prefer coordinates’ selected. . . . .	34
4.6	Location Fragment with a location selected. . . . .	35
4.7	Mapmode Fragment’s view displaying four pre-selected locations. . . . .	36
4.8	Server subsystem’s interactions with the database and app. . . . .	37
5.1	Final product login view. . . . .	43
5.2	Final product register-user view. . . . .	43
5.3	Location permission alert shown when user first logs in. . . . .	43
5.4	Extended saved-location view. . . . .	44
5.5	Saved-location dialog fragment. . . . .	44
5.6	Population-Density Activity’s successful login toast. . . . .	44
5.7	Mapmode displaying the optimal start-time and estimated contacts. . . .	45
A.1	Control for number of contacts in TEST-2, created using excel. . . . .	60

# List of Tables

2.1	List of sensors found in smartphones and their market prevalence [1]. . .	6
2.2	Comparison between location-tracking services [2]. . . . .	9
2.3	List of third-party server solutions and their features. . . . .	12
4.1	‘Users’ table field descriptions. . . . .	41
4.2	‘Location data’ table field descriptions. . . . .	41
5.1	Smartphone diagnostic test results. . . . .	48
A.1	Input data for TEST-2. . . . .	60

# Glossary

Listed below are abbreviations used throughout this report.

- API - Application Programming Interface
- AR - Augmented Reality
- CPU - Central Processing Unit
- GPS - Global Positioning System
- GSM - Global System for Mobile Communications
- GUI - Graphical User Interface
- HTTP - Hypertext Transfer Protocol
- IoT - Internet of Things
- JSON - JavaScript Object Notation
- OS - Operating System
- PEEPS - Population of Environment Evaluation and Prediction System
- RAM - Random Access Memory
- SQL - Structured Query Language
- UI - User Interface
- URL - Uniform Resource Locator

# Chapter 1

## Introduction

### 1.1 Background to the Study

With the advent of the novel Coronavirus appearing towards the end of 2019, population prediction mechanisms have suddenly become commonplace in the minds of the public. With a virus that is highly infectious, particularly in locations with high population density [3], it is in the interest of public health to minimise excessive contact with potentially infected persons. While this pandemic has brought the health benefits of population evaluation to the public consciousness, the potential for population prediction to improve personal and general prosperity has long been available. It is as important for consumers to determine which shopping centres have shorter queues as it is for a new business owner to know what is causing the changing traffic levels in their store. Hence, both population evaluation and population prediction have the potential to revolutionise how we predict prosperity and enhance our well-being.

Large scale population data tracking has become increasingly viable in the last decade due to the inclusion of precise positioning sensors on smartphones and the extensive prevalence of smartphones. Population evaluation mechanisms no longer require third party cameras and beacons, but instead favour crowd-sourced data uploaded from mobile devices. The most successful example of this kind of mass population evaluation and prediction is the Google Maps smartphone app. Maps uses mass user-data to predict live population counts of roads, businesses, and public locations. This type of population tracking does, however, come with a healthy amount of concern for privacy. With the great benefits of population prediction comes the responsibility for preventing bad actors from abusing this information to commit malicious activities. Hence, population evaluation systems need

to focus their efforts on increasing public trust while simultaneously solving problems using this powerful tool.

## 1.2 Objectives of this Study

Due to the immense benefit of population evaluation and the ability for it to be integrated into many applications, this study aims to develop a modular framework with design aspects that can be incorporated into new and existing projects. This framework will take the shape of a smartphone application with integrated network and database connectivity. The app will implement various mechanisms of population tracking, evaluation and prediction which will be complemented by a robust server system.

This project will provide documentation on the design process for these population evaluation mechanisms and conduct testing on the functionality of the system. Additionally, investigative research will be conducted on the system to provide future developers with insight into the operation of such a population evaluation tool. This project emphasises modularity and system-focused design to allow for future developers to choose which aspects of population evaluation they wish to implement. The program will be documented thoroughly for the same purpose.

## 1.3 Scope and Limitations

The scope of this project is limited to the design and research testing of a typical population evaluation tool. So, no large-scale population testing will be conducted, but instead the individual systems of the PEEPS framework will be evaluated to determine their functional parameters. Likewise, the application will be tested using emulators and one typical modern smartphone and as such, may not be representative of how the system will react to all devices.

This project will also be limited to the resources available at the height of the COVID-19 pandemic. This means laboratory hardware and resources will not be utilised nor will testing be conducted on individuals not involved in this research. Since PEEPS's objective is for it to be used as a framework, it should not be viewed as a publish-ready application, as the program's focus will be on creating systems that can be incorporated into other projects, rather than being a completely fleshed out application.

## 1.4 Plan of Development

The project was initiated by performing a thorough analysis of the landscape of app development and population prediction mechanisms. This review was used to formulate the design concepts outlined in the methodology. These design concepts were then finalised and development on the PEEPS server system commenced. This included construction of a robust database and web interface. Once construction of the server was complete, development on the smartphone app began. The development on the app employed the use of a smartphone emulator to allow for swift and concurrent development and functional testing. After the finalisation of the PEEPS application, functional and research tests were devised and conducted to determine the operating characteristics on the completed system. Finally, conclusions were drawn based on the results of the tests and recommendations were formulated.

## Chapter 2

# Smartphones as Population Density Trackers

Modern smartphones provide unparalleled opportunities for people to access ever innovating services. This in turn creates a volatility that makes the mobile landscape of the future difficult to predict and many are unsure what form it will take even in the near future. This chapter aims to provide an understanding of this industrial landscape and outline the demand and technical description of crowd-sourced mobile applications and their ability to analyse and predict population density.

### 2.1 Smartphone Access and Development

The rise in competition within the low-priced smartphone market from brands like Huawei and Samsung have created a revolution in South Africa in terms of Internet access. As of 2019, over 91% of South Africans have access to a smartphone and all the global connectivity that accompanies it [4]. This combined with near total 3G network coverage [4] means that almost all South Africans have become part of the online smartphone application ecosystem. Cheap data-based communication applications like WhatsApp and Facebook Messenger have largely replaced SMS, and revolutionary services like Uber and UberEats have had a similar effect on the taxicab and food delivery industries. Contemporary smartphone applications have sculpted our way of life and this trend is accelerating as we increasingly understand the type of advancements the smartphone revolution has in store for us.

### 2.1.1 App Development

Android and iOS app development has never been easier than it now is, with the release of first-party toolsets like Android Studio and Xcode which allow anyone with the slightest programming background to start working on their application idea. Access to a global market where the barrier for entry for businesses is low and the potential customer base is in the billions [5], has made smartphone application development a priority for companies wanting access to this \$311.25 billion market [6].

South Africa is no exception to this economic surge, having over four thousand apps available on the Google Play Store with nearly 200 million downloads across them [7]. Unfortunately, expensive data prices are a major obstacle preventing South Africa from reaching its potential in harnessing the economic benefits of smartphones. South Africa is ranked 148th out of 228 countries in terms of mobile data prices[8], which may not seem that prohibitive when compared to the United States (188th) and New Zealand (180th) [8], but doesn't take into account the large proportion of the population who can only purchase data in smaller packages. Data when purchased in the 10s of Gigabytes may not be that bad a deal, but South Africa's large impoverished population is forced to buy data at disproportionately expensive prices due to mobile data's exploitative pricing structure. Despite prohibitive data price-structures, the number of smartphone application downloads is likely to accelerate moving forward, due to the Competition Commission's order on mobile-service operators to lower their exorbitant data prices. The implementation of this order has created an average price drop of 33% across all affected data packages [9]. The continued adoption of smartphones and lowered data prices is likely to accelerate South Africa's application development marketplace into a thriving economic sector in the near future.

### 2.1.2 Smartphone Technology

A smartphone's convenience is in a large part due to the considerable number of sensors at its disposal. While the number and types of sensors vary from phone to phone, there are several sensors that are seen as necessary, by the market, for a smartphone to feel complete.

Many newer smartphones tend to market themselves either on the fidelity of their sensors, often in the case of cameras, or the existence of a newly added one. An example of companies using newer sensors to sell devices is the iPhone 12 Pro's incorporation of a



Sensor	Function	Prevalence
Accelerometer	Detects acceleration and vibration.	High
Ambient Light	Determines ambient light levels.	High
Barometer	Measures air pressure.	Low
Bluetooth	Used for data exchange over short distances.	High
Camera	Used to take optical images.	High
Cellular	Used to connect to cellular networks.	High
Fingerprint	Biometric sensor used to verify user's identity.	Low
GPS	Determines geographical position.	High
Gyroscope	Detects orientation and rotation.	High
Heart Rate	Detects user's heart rate.	Low
Magnetometer	Detects magnetic fields.	High
Microphone	Detects and records sound.	High
Proximity	Uses infrared light to detect proximity to device.	High
Thermometer	Monitors internal temperature.	High
Touchscreen	Capacitive sensors that detect interaction with screen.	High
Wi-Fi	Used to connect to local Wi-Fi hotspots.	High

Table 2.1: List of sensors found in smartphones and their market prevalence [1].

LiDAR sensor to be used for more immersive AR experiences [10]. The sheer number of sensors in smartphones provides developers plenty of opportunities for creative methods of solving problems. This can mean that for determining the smartphone's location, an application could use a combination of GPS, Cellular, Wi-fi and Bluetooth sensors to create a moderately accurate approximation of the user's current position.

## 2.2 Location Tracking

Location tracking refers to the ability of devices to use sensors, like GPS and Wi-Fi, to determine their geographical location. With the popularisation of the smartphone over the last two decades, and its increasing reliance on a multitude of built in sensors, apps installed on these devices have unparalleled access to location tracking services. Some of these services are hard to avoid given how much power they give to the user in terms of services. Services like Google Maps allow the user not only to determine their own location, but also use data from millions of other users to construct an accurate approximation of the population densities of individual locations. This section will explore the location tracking methods relevant to this project as well as contemporary examples of services that implement these tracking mechanisms.

### 2.2.1 Methods of Tracking

There are two primary mechanisms of determining your location via mobile connection: position-tracking services and position-aware services [11]. Position-tracking services use an external party's resources to track and maintain the locational data relevant to the user, while position-aware services use the user's device's own diagnostic capabilities to determine its location [11]. Both services have their own advantages, but are typically implemented in different scenarios.

PocketFinder is an example of a position-tracking service that uses a small GPS beacon to determine the location of a loved one or pet. The beacon receives telemetry data from GPS satellites and nearby Wi-Fi signals and uses this data to triangulate the device's location. This data is then displayed on one or more users' mobile devices allowing them always to know the location of whoever has the tracker. This method of location tracking has the upside that it does not rely on any of the phone's built-in sensors. This protects against device failure and allows the tracking capabilities to surpass the capabilities of the mobile device. However, the cost of the method is steep, requiring external tools to do the work that a mobile device is more than equipped for.

While most location tracking services use a combination of cellular, Wi-Fi and GPS sensors, there are plenty of additional sensors which can increase the accuracy of population tracking services. Bluetooth surveillance is a method of tracking users' locations that relies on the built-in Bluetooth sensors of each user's phone [12]. It generally involves the placement of a beacon which periodically scans for nearby Bluetooth devices and uses this data to approximate how many individuals are nearby. While newer versions of Bluetooth drivers allow for some amount of anonymity while keeping Bluetooth scanning on, if this 'hidden mode' is not active, these beacons can be used to determine the owner of the phone being scanned. This creates plenty of ethical concerns related to user privacy and the commercialisation of user-data. Even phones themselves could be used as dedicated Bluetooth tracking beacons, scanning nearby devices for user information. This type of location tracking was not implemented by this project because of the ethical consequences of such tracking mechanisms.

### 2.2.2 Existing Services

While smartphone apps like Google Maps, Waze and Apple Maps present the obvious benefits of location tracking, there are plenty of apps that use this data to provide the

user with services that are unique to the gathering of location-data. Peg Log is such an app, created to perform psychological research using location data gathered from study participants [13]. Since location data has been proven to be predictive of both a user's individual differences [14] and the possibility of depressive behaviour [15], Peg Log could be used as the basis for an app that can prevent self-harming from occurring with individuals.

Find My Friends is an app that uses location data to promote safety and security amongst its users. The app allows you to share your location in real-time with specific people. This implementation of location tracking provides users with safety measures when in unknown environments, or with new people. Such an app is particularly useful in the South African context, where anxiety around the possibility of a crime being committed can be reduced by having others know your location.

The success of mobile games, like Pokemon Go and Ingress, shows that location data has a lot to offer to the field of augmented reality (AR). AR is an interactive experience that mixes our real-world perceptions with augmented sensory information provided by computing devices [16]. Location data is used in the examples above to provide a virtual overlay of our geographic location and supplement it with interactive experiences. Because location data, in this instance, is used as a base from which an experience can be manufactured, the possibilities for its application are extensive. One such application could be the implementation of location-based AR in simulated environments, like for military training, or allowing a user to explore distant locations with real-time movement.

### 2.2.3 Location Tracking APIs

Phones have plenty of tools at their disposal capable of determining their location, from using sensors like GPS, GSM and Wi-Fi to systems that manipulate data gathered from temperature and humidity sensors. Services like the Google Maps API tend to use a hybrid of these systems to minimise resource cost and maximise accuracy. The resulting robustness of Google's location services has made it the de facto method for location integration with over 2 million websites using the API [17]. While it does allow companies \$200 of free usage a month, beyond that you will need to pay a premium to continue having it integrated into your service. This makes the Google Maps API an enticing option for those wishing to quickly implement location services, but for large position-tracking services, it becomes a cost sink that will keep expanding with the userbase.

Service	Selling point	Price
Google Maps	Robust positional tools. Simple mobile application integration.	\$200 free monthly requests then PPR (pay-per-request)
OpenLayers	Offers basic maps cheaply. Has all the basic map functionality. Open Source.	Free
TomTom	Everything offered by OpenLayers, but with strong navigational services.	2,500 free monthly requests then PPR
Mapbox	API of choice for Facebook and Snapchat. Can use custom-made maps.	50,000 free daily requests then PPR
Here	Extensive map visualisation options. Public transit data.	250,000 free monthly requests then PPR
Mapfit	Extreme locational accuracy. Can provide inner-building detail.	50,000 monthly non-commercial requests then PPR

Table 2.2: Comparison between location-tracking services [2].

While there are many alternatives to the Google’s location-tracking behemoth, each has its own positives and trade-offs. OpenLayers, in particular, provides newer companies with a service that can be implemented easily with no cost scaling besides server maintenance. Since it is built on JavaScript, it is possible to create a webapp with fully functioning location services without propriety APIs. While webapps do miss out on a multitude of smartphone specific functionality, their ability to function on any internet connected device makes them highly scalable. Mapbox also provides an enticing alternative to Google Maps for newer developers. Mapbox’s custom-map functionality provides developers with an innovative tool to try new ideas while allowing them a plenitude of free daily requests.

### 2.2.4 Accuracy and Reliability

Services implementing location tracking on smartphones all have a similar issue with the accuracy of the readings being reliant on the GPS, Cellular and Wi-Fi sensors. As a result, the accuracy of the location gathering ability of smartphones will differ greatly. While smartphones with better sensors will produce more accurate results, the fidelity of a particular smartphone’s sensors is difficult for applications to determine. The accuracy can also depend on how populated the area being surveyed is, or on how many large structures are nearby [18].

A study testing the accuracy of iPhone 6’s location discovery ability found that the aver-

age error in horizontal position to be in the range of 7-13 meters [18]. While this number appeared consistent over the course of a year, the accuracy did improve temporarily in the afternoons. If we take this number as a baseline for the accuracy of smartphone location services, the problem of using smartphones as highly accurate location beacons is apparent. A variation of 13 meters could place an individual on the wrong road or in the wrong house which could hamper some location services' ability to provide accurate information to the user about their surroundings. The accuracy of a user's location could be increased, however, by increasing the number of readings and the time over which they are taken. This leaves open the opportunity for apps that require accurate readings but which can be taken over the course of a couple of minutes or hours.

## 2.3 Data Acquisition

Our collective overreliance on the use of smartphones has led them to become an extension of our own thoughts and desires. They store our calendars, our shopping lists, our photos, and our private conversations. In this, they also store desirable data for companies to be used in various applications including revenue collection through advertising, and diagnostic data used to improve their own applications. This enticing amalgamation of information has led to an explosive number of free smartphone apps that use the user's personal information and telemetric data as payment for their services. This has placed consumers in a predicament where they must choose between their privacy, or anonymity, and the countless number of useful applications which will inevitably raise their standard of living.

### 2.3.1 Crowd Sourced Data

The increasing global popularity of smartphones has made them prime conduits for sourcing large amounts of unique data points. Each app download is not only a potential revenue stream for its owner, but also a contributor to a wider network of information. This has led to the rise of crowd-sourced applications which can be found in abundance on both Android and iOS app-stores.

Crowd-sourcing is defined by Merriam-Webster as "the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people" [19]. This definition indicates that there are a few key criteria that a smartphone app needs to meet

in order for it to implementing crowd-sourcing. Firstly, it must be collecting data that is being used to provide a service to its customers, itself or a third party. Secondly, it must have a large enough install-base so that the data gathered is useful. While the first aspect is easy to implement, the success of any crowd sourcing application depends on its ability to accrue a large enough user base for the data gathered to be useful. An app which reviews destinations, for example, is only useful if it has enough user reviews for the app to provide meaningful reviews to its audience.

An example of a successful crowd-sourcing smartphone application is Zomato, which crowd-sources restaurant reviews and uses them to help customers choose where to eat. With nearly 10 million reviews, Zomato's large user-base provides both the company and its users with diverse and statistically significant information on where to buy food. The success of this app also demonstrates the cyclic nature of crowd-sourcing applications. As you accrue more users, the data that you crowdsource is more useful which incentivises more users to join. However, the contrary is also true: if you fail to achieve a high enough user-base, the information you provide will not be useful enough to sustain your audience and, hence, your audience will shrink indefinitely.

Crowd-sourcing apps are therefore incentivised to gather proxies for their desired information to bolster their database. Smartphones' diverse set of sensors, detailed in Table 1, provide the option for apps to find non-conventional methods of accruing data. As an example, the crowd-sourced application Pothole Patrol, developed by MIT for monitoring road surfaces, uses a combination of location services and the phone's built-in vibration sensors to track and report pothole locations [20]. While the acceleration data gathered by the vibration sensors would not normally be used for this type of analysis, smartphone sensors provide developers with the ability to solve design decisions in interesting and innovative ways.

### 2.3.2 Data Storage and Server Solutions

Because of the resources needed to maintain a dedicated server, third party server and database solutions are often employed to jumpstart networked app development. These solutions tend to be very cost efficient for low volumes of data transfer but become costly when the volumes of data balloon over time. Hence it is an important choice between implementing a dedicated server or renting the resources instead.

The solutions presented in Table 2.3 are very similar in their functionality, so the choice

Service	Features	Pricing
Amazon Web Services	High scalability options. Over 76 global server locations. Dynamic and resizable resources.	Limited free tier available. Multiple payment formats to choose from.
Microsoft Azure	Ability to mix local and cloud infrastructure. Machine learning integration.	Some free services. Paid services are pay as you go.
Google Cloud	Google services integration (advertising, search, etc). Android app integration. Machine learning integration.	Pay as you go with free trial.

Table 2.3: List of third-party server solutions and their features.

may depend principally on the pricing structure. However, if data security and authority over your resources is paramount to your goal, a dedicated server may be more suited to your use-case as you would have complete control over its implementation.

### 2.3.3 Smartphone User Privacy

The number of sensors present in smartphones, as shown in Table 2.1, means that this small handheld device gathers a disproportionate amount of information belonging to the user. This combined with smartphones' "always-on" feature and persistent internet connection means that there is plenty of opportunities for users' information to be stolen. Google's Android and Apple's iOS have converged in their response to this potential privacy threat by providing the user with the authority to grant or deny apps' access to important sensors like GPS, and an app's ability to start services immediately after a device has booted [21]. However, these permissions do not cover all types of sensitive data. Android notably does not have permission requirements for apps to connect online or to set phone alarms. Moreover, one of the more egregious of these privacy violations is the possibility of an app to scan the Android device for all packages installed without any permissions. This data could be used to identify both the user of the phone, and particular vulnerable packages which can be hijacked to conduct malicious acts on the user's device[21].

Both Google and Apple have taken steps to prevent data theft by encrypting storage and data transmission, but these methods of protection still have gaps in their security strength. Any sort of data protection implementations by these OS manufacturers are however no replacement for user vigilance. Should the user allow a particular app permission to access their sensory data and accept the app's terms and conditions, there is

not much one can do to prevent their taking and analysing personal information. The only solution for the user would be to uninstall the application.

Location information constitutes some of the most sensitive and valuable data gathered by smartphones. Besides determining your current location, this data can be used to calculate your identity, place of residence, work, diet, and social circles. This makes locational data both important for legitimate apps like Uber and Maps, while also being incredibly valuable to third-party advertising companies and malicious actors like scammers. In 2015, the Pew Research Center studied the habits of smartphone users and found that 90% of smartphone users use apps that rely on locational data, up from 74% in 2013 [22]. However, even if you turn off your Android devices location services, Google will still intermittently upload your location to their database, making complete anonymity impractical without considerable effort [23].



# Chapter 3

## PEEPS Development Methodology

### 3.1 Problem Definition

The aim of this project is to provide valuable insight into smartphone development and location tracking services by designing a crowd-sourced population tracking smartphone application. This project will deliver both a fully functional open-sourced Android application, and investigative testing into the ability of this application to determine location-population density reliably, accurately and with high scalability. This application is intended to be used as a framework for further development, providing developers with relevant groundwork and insight to create crowd-sourced and population-integrated smartphone applications. The deliverable is not designed to be an API that can be incorporated into other projects, but rather an example of a working population tracker with its bounds and abilities clearly defined. Future developers will not need to incorporate every aspect of the PEEPs application, like crowdsourcing or server hosting, for its framework to be relevant, as the modular nature of smartphone app development allows for each subsystem to be programmed, tested, and incorporated independently.

The planning for this project was initiated by developing a list of fundamental requirements which serve as a baseline for an acceptable project deliverable. These requirements determine whether the application has adequate functionality, whereby research testing can be conducted to gain insight into the bounds of crowd-sourced population trackers. Furthermore, project specifications were devised to limit the scope of the project and focus development on the most important aspects of population tracking in smartphone apps.

## 3.2 Development Strategy

Due to the investigative nature of the PEEPS application, a design approach needed to be formulated that would put an emphasis on the creation of a well-defined deliverable. Subsequently, this deliverable would undergo thorough tests to determine its abilities and shortcomings. This linear and sequential design approach is best implemented using the Waterfall Model. The Waterfall Model treats each development phase as independent, with the completion of one phase leading into the inception of the next. Because of the lack of design iteration conducted while implementing this strategy, the Waterfall Model performs best when the project's requirements are well documented. This is the case with PEEPS as the goal is to produce a well-defined product that can then undergo thorough testing. Any shortcomings of the app discovered by the testing phase are not design failures but instead insights into how the app could better be implemented. Likewise, the successes of the app serve as useful observations for others wishing to implement similar functionality into their own app.

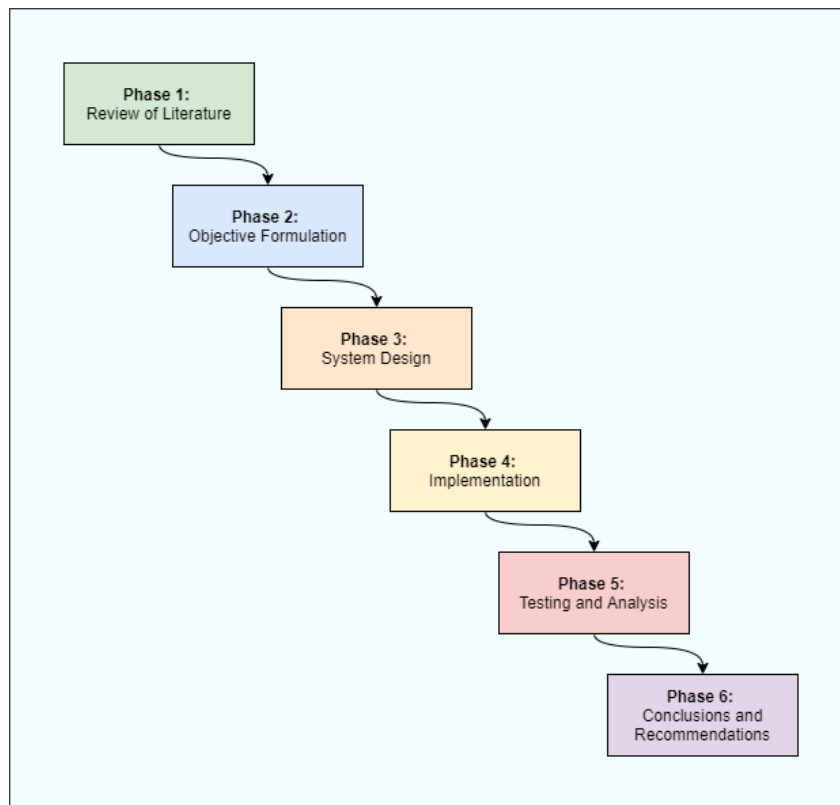


Figure 3.1: Project phase breakdown using the Waterfall Model.

The rigidity of this type of developmental strategy has a number of advantages and disadvantages which will need to be considered while implementing it. The most valuable advantage of this method is the simplicity and specificity of its different phases. Each phase has set objectives to be accomplished and, once they have been achieved, the

next phase begins. It also means that the product testing does not require revisiting the implementation, allowing for well documented and coherent results. However, the rigidity of the Waterfall Method does come with shortcomings which make it non-ideal for certain types of projects. Since the implementation is started late in the developmental cycle, it is unlikely that the quality of the design can be verified until it may be too late. The lack of design iteration during the implementation phase also makes it difficult to gather information about what implementing the design may entail, which will stunt the design phase. Luckily, due to this project's clearly designed objective, the disadvantages posed by the Waterfall Design Method are unlikely to hamper the project's development.

### 3.3 Project Requirements

The project requirements are the minimum necessary criteria for which the deliverable can be described as a successful implementation. Likewise, completion of these requirements is necessary for this deliverable to be able to undergo sufficient testing.

1. For the PEEPS application environment to be considered testable, it should:
  - (a) contain a smartphone application that can be installed on the latest version of its relevant operating system,
  - (b) implement a database structure that will be used to store user and location data,
  - (c) provide server software that will act as an intermediary between the user and the database.
2. For the PEEPS smartphone application to be considered testable it should:
  - (a) allow users to create accounts and log into the application to ensure user data security and privacy,
  - (b) allow the user to select a geographical location and the application will then provide relevant information about the population statistics of that location,
  - (c) routinely upload the user's geographical coordinates to a server so that it can provide population data about the user's location to other application users.
3. For the PEEPS database to be considered testable it should:
  - (a) implement a table which stores users' login details,
  - (b) implement a table which stores location data with timestamps.
4. For the PEEPS web interface to be considered testable, it should:

- (a) be able to receive user data from the smartphone application and forward it to a local database,
- (b) ensure user-data and database fidelity.

## 3.4 Project Constraints

Due to the application being developed during the COVID-19 pandemic, the list of constraints reflect the necessity to mitigate the chance of infection from interactions with potentially infected individuals. While these constraints do hamper the ability of the application to undergo user testing, due to the investigative nature of the project and its reliance on software, this will not affect the ability of the PEEPS framework to undergo thorough technical testing. The list of constraints is as follows:

- Zero reliance on physical components besides the computing devices used to run the software.
- Zero tests of the application framework involving non-developers.
- Number of instances of the app being run, limited by smartphone access by researchers.
- An internet connection is required for the application to perform.

## 3.5 Technical Specifications

These technical specifications detail the more specific aspects of the deliverable and, while not necessary for the project to undergo testing, are what this project aims to deliver.

### Mobile Application Specifications

#### SA-1: General Specifications

SA-1.1: App must be able to run on any version of Android between 6 and 10.

#### SA-2: Graphical User Interface

SA-2.1: Must follow a consistent colour pallet with a primary and secondary colour selection.

SA-2.2: All user interactions with the app must be clearly interpretable from the GUI.

SA-2.3: Buttons must be placed below text inputs in each activity or on the toolbar.

#### SA-3: Location Services

SA-3.1: Location must be uploaded to server at regular intervals.

SA-3.2: Location upload interval must be shorter than 20 minutes.

SA-3.3: Location data is uploaded when app is minimised.

SA-3.4: Location data is uploaded when app is closed.

SA-3.5: Location data upload starts on smartphone boot if user has logged in previously.

#### SA-4: User interaction

SA-4.1: User will be able to save 'favourite' locations by entering their coordinates or address.

SA-4.2: Saved locations will be stored locally using an SQL database.

SA-4.3: Saved locations will display current population density status on the app's main page.

SA-4.4: When selected, saved locations will display a graph showing the level of activity at that location for that day, as well as the predicted activity for the rest of the day.

#### SA-5: Data security

SA-5.1: User will need to login with correct username and password to access application.

SA-5.2: No population density data displayed on the app will contain information about the user who uploaded it.

### **Database Specifications**

#### SA-1: General Specifications

SA-1.1: Database will be secured with appropriate password.

#### SA-2: Tables

SA-2.1: Database will contain a table storing users' login data.

SA-2.2: Database will contain a table storing users' geographical data.

SA-2.3: Location table will require timestamps for each location stored as well as the username of the user who uploaded it.

### **Web Interface Specifications**

#### SA-1: General Specifications

SA-1.1: Server is accessible from devices outside its home network.

#### SA-2: Data Communication

SA-2.1: User and geographical data sent to the server must be checked for database compatibility.

SA-2.2: Data sent to the server, if valid, will be stored on the connected database.

SA-2.3: All location data processing will be done by the server, and only results will be forwarded to the application.

#### SA-3: Security

SA-3.1: Data retrieved by the app from other users via the server must be anonymous.

# Chapter 4

## Design and Implementation

This chapter will detail the design and implementation of the PEEPS framework. It begins by discussing the various design proposals which developers should take into consideration when developing an app of this type, and then using these proposals it will justify the final design of the app and server components. Subsequently, this chapter will discuss the system level view of the project before detailing the finer aspects of each subsystem within the PEEPS framework.

### 4.1 Conceptual Design

Due to the malleable and open nature of software engineering, it is useful to take aspects of each system and interrogate how it is best to approach them. This chapter will detail that process by looking at the options available for each major system concept and provide insight into how different approaches can take the application in different directions. This chapter will also justify why particular options were chosen over the alternatives.

#### 4.1.1 Concept Design Proposals

##### A. Smartphone Application Operating System

For mobile application development there are always two operating systems that need to be considered: Google's Android and Apple's iOS. Combined, these two mobile operating

systems make up 99.42% of the global smartphone market [24] and applications for each of these operating systems are unique both in their design and development strategy. The choice of which of these OS's to target determines the application's audience, development software and documentation resources. As a result, this decision is best made early in the development lifecycle. Each operating system also has its own development language and first-party development applications that allow for a seamless development strategy.

Proposals:

1. **Android, programmed using Java:** Android devices make up over 74% of global mobile smartphones, making android development the most beneficial in terms of potential reach [24]. Google released the development environment Android Studio in 2014, which provides a variety of application frameworks for jumpstarting development. The official language for Android app development is Java which has a large backlog of libraries and other resources from its 20+ years of existence, making it ideal for both new developers and experienced veterans.

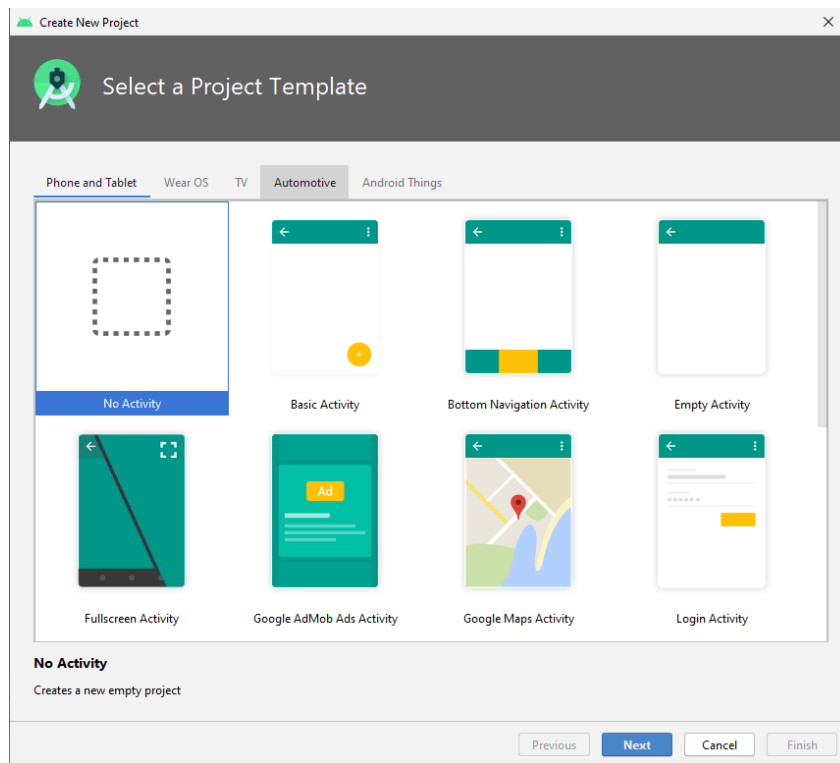


Figure 4.1: Android Studio's activity templates used to jumpstart development.

2. **iOS, programmed using Swift:** While iOS devices are less common globally, their large market share in the US always make them an option worth considering. Unfortunately for the scope of this project, it is much harder to find enough apple devices for mass testing due to android smartphone prevalence. Swift is Apple's



first-party programming language unveiled in 2014, making it much younger than Android's alternative. Due to its youth, the developers of Swift have drawn techniques from established programming languages to create a modern language that is learner-friendly and robust. However, this comes with the downside of a lack of available documentation and troubleshooting resources. Swift's support from Apple does mean that it is tailored to iOS application development, making it a tempting choice for new developers who want a programming language tailored to them.

3. **Android and iOS, programmed using Flutter & Dart:** Flutter is Google's foray into programming language development. Its purpose is to bridge the gap between Android and iOS development, providing an engine that can compile a codebase for both platforms. Flutter uses Dart, an object orientated programming language also developed by Google, to write programs compatible with both platforms. Much like Swift, Dart is relatively new, making its documentation and troubleshooting resources more limited than alternatives. Fortunately for Flutter, its dual operating system approach makes it extremely valuable when one's goal is to develop for both platforms. Unfortunately, because Flutter's objective is to be used to create for both platforms, it still relies on Java and Swift for some background service functionality.

## B. Database Type

While the choice of database will have some impact on how data will be stored from the smartphone application, most databases are very similar in their makeup, and choosing one is more of a preference than it is a significant design decision. What will change between databases is the various extensions that can be used, the databases' available datatypes, and their method of interacting with web interfaces.

Proposals:

1. **MySQL:** MySQL is one of the most popular databases for a variety of reasons: it is free, has extensive functionality and has plenty of user interfaces that can be chosen from. By using the free version, you will be missing out on some of the more advanced features, but this is offset by the sheer robustness of MySQLs database implementation.
2. **SQLite:** SQLite is a versatile database that is great for quickly creating databases that are reliable, portable, and lightweight. However, it has disadvantages of not being built for high levels of HTTP requests and of having a database size limit.

These factors make it not as desirable as alternatives for applications that need high scalability.

3. **PostgreSQL:** PostgreSQL is a database implementation that has plenty of advantages over its competition. It allows for both structured and unstructured data, is highly scalable, supports JSON, and has plenty of available extensions. A relevant extension to this project is PostGIS which allows PostgreSQL to handle complex geometric data.

## C: Web Application Software

To interact with a database run from a server over HTTP, you will need to implement some form of web interface. This can be done by developing software designed to interact with incoming HTTP requests. This type of web development is typically done using the PHP language, but Python has made itself a compelling alternative with its functionality extending beyond just web applications. For purely web development, both can perform the job, but the choice between them may rest on how much you value Python's large suite of unique libraries over the more established PHP landscape.

Proposals:

1. **Python:** It is assumed amongst software engineers that for any project, there is usually a python library or framework that could help you accomplish it. Web development, in this case, is no exception. With the introduction of the Django web framework, Python has made itself a strong alternative to the more established PHP. The sheer versatility of this programming language means that when using Django, you have access to far more functionality than just web development. Additionally, Python has a simple, but elegant syntax that can be easily understood by newer developers.
2. **PHP:** PHP is a language used strictly for web development. Its popularity in the web application space means that it's the language of choice for large web-based companies like Facebook, Wikipedia and Tumblr [25]. Its 'C'-style syntax means that it is easy to pick up for developers new to the language, and its persistence since its inception means that there are plenty of resources available to you as you develop your application. While Python is more popular than PHP for general use applications, PHP is still far more popular when it comes to web applications [25]. The choice between them may come down to whether you plan to make use of Python's large number of applications or stick to purely web development.

### 4.1.2 Final Design Decision

This section details the design choices made between the options presented in Section 4.1.1. Note that for any combination of mobile OS, database type and web application software, insights gained from this project will still be relevant. Each proposal option is one that could be justified, and which is most appropriate for your own project will be dependent on your developmental goals and perspective.

**A.** The Android mobile operating system was chosen as the designated development platform for this project. Android was chosen primarily for its greater reach globally, making the results of the project more relevant to the average developer. Additionally, Java has far more available resources in terms of forum interactions and tutorials, speeding up the development process for complex systems like background processes and location-tracking activities. This decision is the one that has the least transferable results between related population density mobile apps, as it only takes into consideration the Android use-case and ignores the differences in results that will be caused by developing for iOS instead. The operating systems are similar enough where the insights gained will still be relevant, but iOS developers should be aware that not all results may map exactly to Apple's ecosystem.

**B.** PostgreSQL was chosen as the preferred database for this project. The PostGIS extender for PostgreSQL allows for seamless geographical object interaction which is perfect for this project's use-case. This makes PostgreSQL the clear choice over its alternatives. If developers were to use other database types, locational data could instead be stored as vector types or one could split the latitude and longitude values into individual float values. Developers using different databases should be aware that much of the interaction between the web application and the database is easily transferable between databases, and the difference in performance testing should be minimal.

**C.** The web application language chosen for this project is PHP as the scope of the server software has no need for the extended capabilities of the Python language. Since PHP is more commonly used, the development insights and codebase should be more applicable for developers wishing to use the PEEPS application framework. Should developers want to expand the capabilities of their web application, Python is the recommended solution.

With the main proposals for the PEEPS application framework finalised, it is possible to start designing the subsystems that will act as independent modules capable of being incorporated into any population tracking application. These subsystems will still be relevant for developers choosing alternative programming methods, as the discussion will

still be at a high level of abstraction.

## 4.2 The PEEPS Application Structure

With these design decisions justified, it is now possible to explain in depth the system level analysis for this project. This section will detail the system level design decisions taken when developing the application to be run on mobile devices.

### 4.2.1 System Level Overview

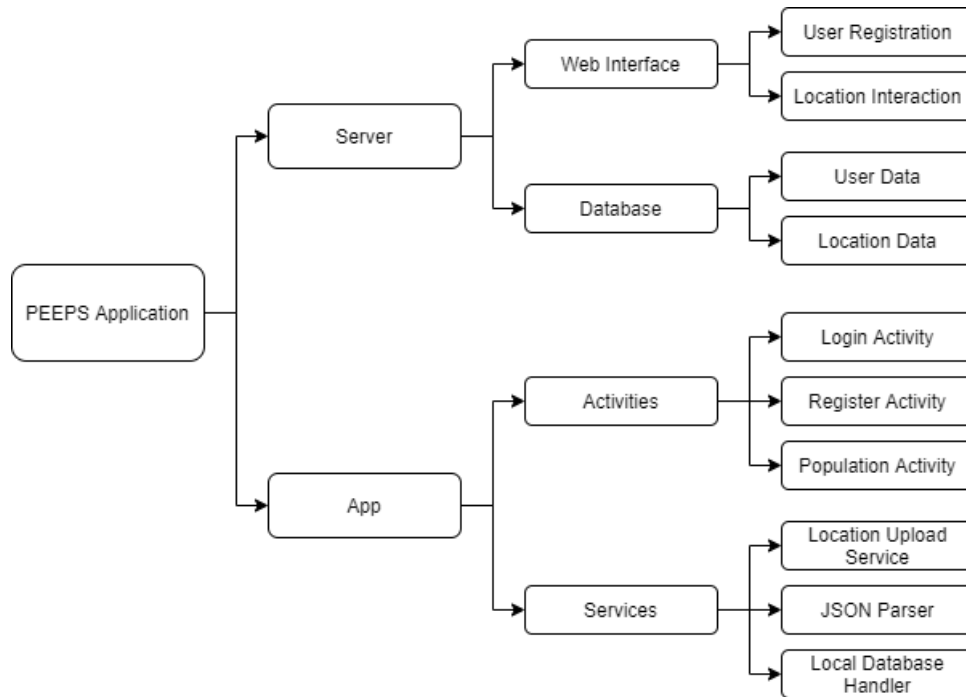


Figure 4.2: High level overview of the main application systems and their components.

The design of the PEEPS framework is split into two independent systems: the smart-phone application and the server. Each of these systems is independent in the sense that one could be swapped out with little change in the operation of the other system. The application could upload to a different server type or the server could receive data from a different application and the fidelity of the overall framework will remain operational. The inter-system interactions will be summarised in Section 4.2.2 and will be discussed in more detail in each subsystem’s section.

### 4.2.2 Inter-System Interaction Summary

Since the server system acts as the ‘hub’ for all the stored user and locational data, it will only interact with the mobile application when information is requested by the application. This will happen when two conditions are met: either the application wants to retrieve or store user login data, or the application wants to retrieve or store geographical data. When this is not happening, the server will remain dormant, while still being accessible by backend managers who wish to interact manually with the data. Likewise, the smartphone application can still run without a connection to the server, although its functionality will be severely reduced.

### 4.2.3 Data Anonymity

Because of the sensitive nature of population tracking and the possibility of abuse, all data displayed to the user by the PEEPS system is predicted from past data and is completely anonymous. Should live population data be displayed, malicious users could abuse the system to perform illegal acts such as theft. Therefore, the PEEPS system uses anonymous prediction mechanisms to provide useful insight into probable location-populations without compromising individuals’ or businesses’ privacy and safety.

## 4.3 App Design and Development

### 4.3.1 System Description

The Peeps smartphone application can be split into three main subsystems: The login subsystem, the register subsystem, and the population-density subsystem. The fundamental aspect which separates these subsystems is their unique Java activity which handles the application’s UI and its interaction with the user. The login and register subsystems administer the user’s identification and registration within the PEEPS app. They allow the user to create an account and access the app’s main functionality by logging into the app using that account. When a user has successfully logged in, it will be able to interact with the application’s population-density subsystem. This subsystem provides the bulk of the application’s functionality, which is split into the ‘saved location’ and ‘mapmode’ fragments. These are discussed in more detail in Section 4.3.4.

### 4.3.2 Java Class Structure

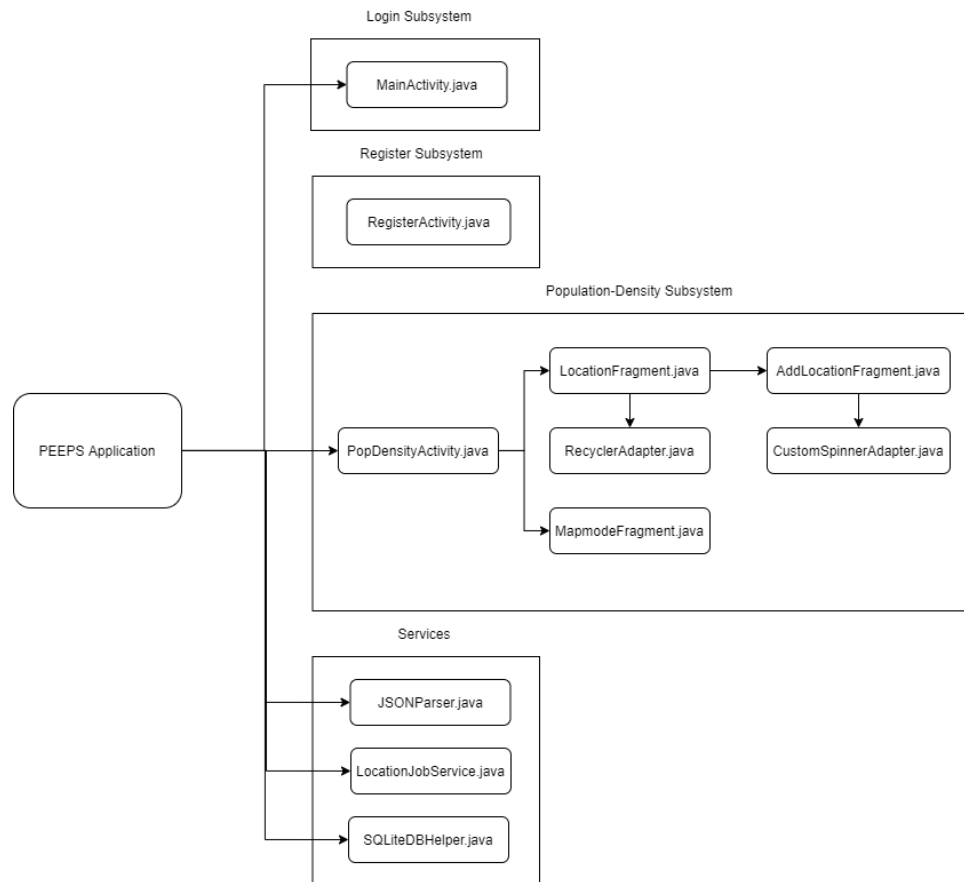


Figure 4.3: Java class hierarchy for the PEEPS App.

Android applications are typically built using a combination of Java class files and xml resource files. The classes act as the method by which the app can receive, interpret, and manipulate information where the xml files are templates which are used by the app to display application artifacts to the user. Because of the modular nature of programming, it is possible to fracture the structure of the application into various types of components. The main components which the application makes use of are as follows:

#### Activities

Activities are the highest-level class in the application hierarchy. These classes manipulate the UI and handle all the interaction performed between the user and the app. For example, when a user attempts to access a new screen on the app, the activity will have to handle this transition. In the PEEPS app, each activity is its own subsystem which provides some unique functionality to the user. The three activities in this application are the login activity, register activity and population-density activity.

## Layouts

Layouts are the blueprints used to create the application GUI. These are coded in xml and are interacted with using their Java class. These layouts can describe either entire activity UIs or individual components of an activity's interface.

## Fragments

Fragments are GUI elements and Java classes which are used by activities to display different UI's without transitioning to a new activity. In this application, Fragments are used by the population-density activity to perform different interactions with the user, each with a unique GUI.

## Services

Services act as supplementary Java classes that provide additional functionality to the application. These can be used by as many activities or fragments as required. The Services being used are as follows:

1. **JSONParser:** This class handles all html requests between the app and the web interface hosted by the server. It is run from a thread separate from the main thread.
2. **LocationJobService:** This class is a background process that intermittently uploads the user's location to the server's database. It is scheduled by the Population-Density Activity to run every 15 minutes and starts on smartphone boot.
3. **SQLiteDBHelper:** This class handles interactions between the activities/fragments and the app's local SQLite database. This database stores information about the user's saved locations.

The operation of these services is expanded upon in Section 4.3.3.

## Adapters

Adapters are used by classes that wish to implement complex GUI elements like recyclers and custom spinners. In this project they are used by the LocationFragment class.

### 4.3.3 Application Services

#### JSONParser

The JSONParser is the class that handles all HTTP communication between the application and the server. Because network communication can take a significant amount of time, this class is accessed by the program using non-main threads and runs in the program background. To use the HTTP functionality of this class, the thread accessing it sends it a URL name, method of transmission and a JSON object with all the data that needs to be transferred to the server.

```

1      String message = jsonInput.toString(); // message to send
2      URL url = new URL(urlName); //server url
3      HttpURLConnection con = (HttpURLConnection) url.openConnection();
4      con.setReadTimeout( 10000 ); //milliseconds
5      con.setConnectTimeout( 15000 ); //milliseconds
6      con.setRequestMethod("POST");
7      con.setDoInput(true);
8      con.setDoOutput(true);
9      con.setFixedLengthStreamingMode(message.getBytes().length);
10     con.setRequestProperty("Content-Type", "application/json; utf-8");
11     con.setRequestProperty("Accept", "application/json");
12     //open
13     con.connect();
14     //send
15     OutputStream os = new BufferedOutputStream(con.getOutputStream());
16     os.write(message.getBytes());
17     //clean
18     os.flush();

```

Listing 4.1: JSONParser sending a JSON variable to the web interface.

When the JSON Parser's makeHttpRequest method is called, of which Listing 4.1 shows a portion, it will attempt to connect to the URL provided and attach the JSON data to the request. If it successfully reaches the web interface, it will in turn receive a JSON file containing the status of the HTTP request from the server.



## LocationJobService

The LocationJobService is a class with two main functions: determining the user's current geographical location and uploading this location to the server. The service is scheduled after login by the Population-Density Activity and will run periodically with specifications set by the Job Service Scheduler. The class initially checks to see if the user has given explicit permission for the application to determine the user's location and then activates the Fused Location Provider Client. This client interacts with Google's battery-efficient fused location API which uses a combination of the various receivers on the phone to pinpoint the phone's location.

```

1      LocationRequest mLocationRequest = LocationRequest.create();
2      //set request parameters
3      mLocationRequest.setInterval(60000);
4      mLocationRequest.setFastestInterval(5000);
5      mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
6      mLocationCallback = new LocationCallback() { //run when location is found
7          @Override
8          public void onLocationResult(LocationResult locationResult) {
9              for (Location location : locationResult.getLocations()) {
10                 if (location != null) {
11                     if (checkPermission()) {
12                         //stop getting location updates
13                         mFusedLocationProviderClient
14                             .removeLocationUpdates(mLocationCallback);
15                         //new thread needed for network access
16                         new Thread() {
17                             @Override
18                             public void run() {
19                                 //send location to database method
20                                 sendLocationData(location);
21                                 //JobService finished
22                                 jobFinished(jobParameters, false);
23                             }
24                         }.start();
25                     }
26                 }
27             }
28         }
29     };
30 
```

```

31     if (checkPermission()) {
32         //start location update request
33         mFusedLocationProviderClient.requestLocationUpdates(mLocationRequest,
34             mLocationCallback, null);
35     }

```

Listing 4.2: Extract from the LocationJobService’s getLastLocation(...) method.

After the fused location client has finished requesting a location update, a new thread will be created (see line 16 of Listing 4.2) to package the location information into a JSON compatible format. This thread then makes use of the JSONParser to send this data to the server.

## SQLiteDBHelper

Of the three services in the application, the SQLiteDBHelper is the only one that doesn’t require additional threads to run it. This service’s task is to set up and monitor the app’s local SQLite ‘savedLocation’ database containing the name, coordinates, and chosen image for the user’s saved locations. This database is accessed by the program by using a cursor that can traverse through entries that are selected by a query. Since the database is stored on the app, it can be designated to have different locations for each logged-in user, although this has not been implemented at the time of this project’s completion.

### 4.3.4 Subsystem Analysis

#### Login Subsystem

When the user starts the PEEPS app, it will initially display the Login subsystem’s activity ‘MainActivity.java’. This activity allows the user to login with a pre-existing PEEPS account or be directed to a page where the user can create an account. This activity determines whether the user’s login information is genuine by using the JSON-Parser service to connect to the server. The user’s username and password are stored in a JSON object which is subsequently sent to the PEEPS server using a HTTP post request. The JSON parser waits for a reply from the server with a new JSON object. This object contains information about whether the request for data from the database was successful and whether there were any users matching the details uploaded. Once

this has been completed, the app will show a message informing the user of the success or failure of the login request. Should the user's login data be correct, the user will then be redirected to the population-density activity and *its* subsystem will take over.

## Register Subsystem

This subsystem works similarly to the Login subsystem in that it will receive data from the user and attempt to send it to the web server. Should the web server receive a valid login username and password from the user, this data will be added to the user-database, a success message will be displayed, and the user will be returned to the login page.

## Population Density Subsystem

This subsystem is the most complex of the three and handles all the location specific functionality of the app. Because of this complexity, the subsystem is split into one activity, PopDensityActivity, and two fragments, LocationFragment and MapmodeFragment. When the activity is called, it creates a bottom navigation bar which the user can use to switch between fragments, and then sets the LocationFragment as the current fragment. Each fragment handles a specific location-based functionality and performs most of the work in the subsystem.

Once the fragments have been set up, the activity starts the location upload process. This is the crowd-sourcing process which uploads each user's location periodically so that other users can be informed of how dense certain areas are, and then use that information to schedule when they will visit a particular location. Because of the potential abuse apps can cause by storing the user's positional data, the user needs to give the app explicit permission to access its location. Hence when the activity is started for the first time, the user will need to give the app permission to access this information. Once the permission has been granted, the application will attempt to start a type of process called a Job Service. This Android service can be used to perform persistent, periodic, and highly specific workloads. The job is scheduled using the following code:

```

1  // schedule new location JobService
2  ComponentName componentName = new ComponentName(this,
    LocationJobService.class);
3  JobInfo info = new JobInfo.Builder(35800, componentName)
4      .setRequiredNetworkType(JobInfo.NETWORK_TYPE_ANY)

```

```

5      .setRequiresCharging(false)
6      .setRequiresDeviceIdle(false)
7      .setPersisted(true)           // persists after reboot
8      .setPeriodic(15*60*1000)     // repeats at 15 min intervals
9      .build();
10     Log.d(TAG, "LocationJob scheduling in process.");
11     JobScheduler jobScheduler = (JobScheduler)
        getSystemService(Context.JOB_SCHEDULER_SERVICE);
12     //schedule job
13     int resultCode = jobScheduler.schedule((info));

```

Listing 4.3: PopDensityActivity scheduling the LocationJobService.

While the timing on the execution of a job service is not exact, the scheduler will try to keep the execution interval as close to 15 minutes as possible. This timing is investigated in Section 5.3.

### 4.3.5 Population Prediction Functionality

The Population-Density Activity provides two different views which the user can use to gather population prediction information. The fragments, which these views are bound to, are the Location Fragment and the Mapmode Fragment. This section explains in detail the functionality of these fragments.

#### The Location Fragment

The purpose of this view is to allow the user to save locations of interest so that their population density can be analysed. If a user wanted to know how busy a shopping mall is, they could enter the coordinates of the mall and the app will save the location to its local database where its population information can be displayed to the user.

Figure 4.4 is what the app will display once the user has added a number of locations to their saved list. The colour of each location card and status message displayed below each location title will change, depending on the number of people predicted to be at that location currently. The bottom toolbar of this screen allows the user to switch between the two available functions of the Population Density subsystem: the saved-locations fragment and the mapmode fragment. The top toolbar displays the name of the app's

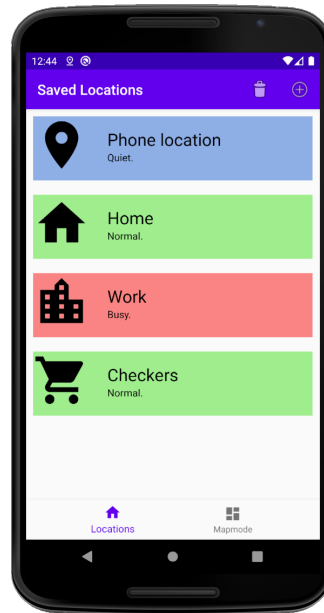


Figure 4.4: Location Fragment displaying a user's saved locations with their population status.

view and two interactable buttons. The first of these clears the user's saved locations and the second allows the user to add additional locations. Should the user select this second button, the app will display a fullscreen dialog-fragment. This is an overlay over the current activity which keeps the activity running in the background while the user enters information. This method of input is preferred over creating additional activities or fragments as it simplifies a significant amount of the application layout at the cost of additional processing power. This location input screen is displayed in Figure 4.5.

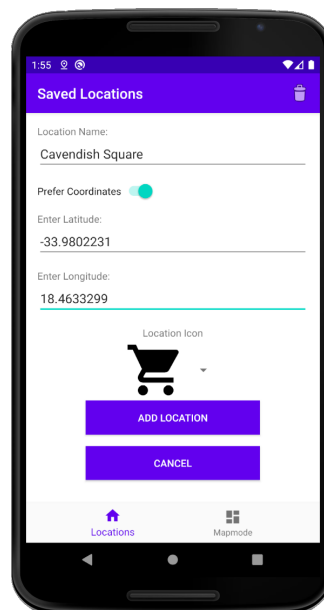


Figure 4.5: Add-Location View with 'prefer coordinates' selected.

The user can use this view to enter both a location's name and coordinates, and then select an appropriate icon to represent the location. A switch is displayed below the name which the user can use to change between entering a location's address or its coordinates. The address functionality is not currently available to the user and acts as a suggestion to future developers of alternative methods of location input. While these figures show the use-case from a consumer's perspective, this functionality can equally apply to business owners, as shown in Figure 4.6.

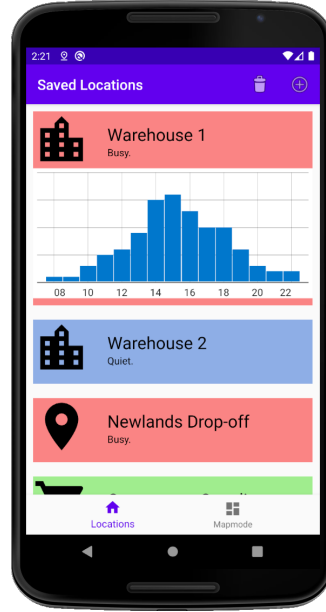


Figure 4.6: Location Fragment with a location selected.

Figure 4.6 is also an example of the Saved-Location Fragment with the first saved-location selected. When such a location is selected, its card will expand to show more detailed information about the predicted population density at that location. In this example, Warehouse 1 is expanded to show what the population is predicted to be throughout the day. Each bar in the graph depicts an hour interval which displays the number of unique users who have previously uploaded their location within 20 meters of the location's coordinates. More detail about how location data is gathered from the server is described in Section 4.4.3.

Since the time in Figure 4.6 is 2:21 PM, the app looks at the 14:00-15:00 interval to determine how busy the location is. In this example, this interval is busy compared to the rest of the day, and the app determines this by comparing the current number of people in that time bracket to the maximum difference in population throughout the day. If the number of people is in the top 30%, it determines the state to be busy, and similarly, the bottom 30% is determined to be quiet. All other states are displayed as a normal amount of activity.

### The Mapmode Fragment

This view is designed as a method of testing the PEEPS framework’s ability to perform complex analysis of population data. It initially selects four locations and determines the time one would spend traveling from the first location to each subsequent location at a walking pace. This timestamp and geographical data is then sent to the server which gathers the predicted population density data at those locations. When this data is received back by the app, it will calculate the optimal time that the user could start their journey in order to minimise contact with other individuals.

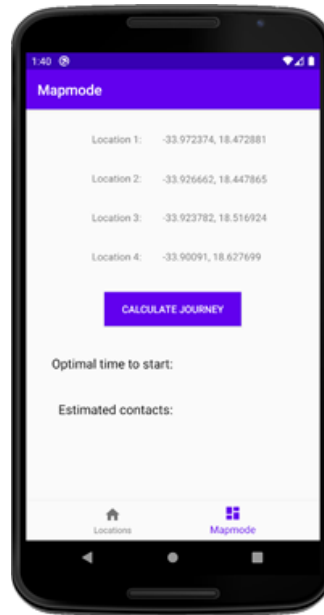


Figure 4.7: Mapmode Fragment’s view displaying four pre-selected locations.

When the user selects this view using the bottom toolbar, they will be presented with four predetermined locations and an interactable button. The coordinates of these locations are set up within the app’s software and are designed to simulate an environment where the user could input them manually. On selection, the button will start the population calculation and, when completed, the app will present the data near the bottom of the view. The contact prediction functionality is meant to be an example of how population data can be used to provide the user with meaningful information that can be used during the COVID-19 pandemic to minimise contact with potentially infected individuals. This data analysis mechanism could additionally be modified to determine vehicle traffic, population hotspots and the user’s likelihood of contracting a seasonal flu.

## 4.4 Server Design and Development

### 4.4.1 System Description

The Server system oversees the handling of all user and location requests coming from the application. While this system was designed to be implemented on a local dedicated server, an alternative method that may be more scalable, depending on the network traffic, will be to implement the server using third-party hosting solutions like Amazon EC2 or Google Firebase. These alternative implementations do require outsourcing of server hardware but in turn simplify the development process. For the scope of this project, a local server provides the precise control needed for research testing while still having the capability to store enough data and do so swiftly.

### 4.4.2 Subsystem Analysis

The two subsystems that the server utilises are the web interface and database subsystems. These act in tandem so that whenever a request to access database data is sent from the app, the web interface will act as a mediator between the app and the database. For example: if the user wishes to register a new account, the app will collect the user's data into a single JSON variable and forward it to the address of the web interface's *user\_login.php* file. The web interface will then use this data to query the database in order to determine whether there is a conflicting username. Should there be no conflict, the web interface will update the database with the new user's details and return a new JSON variable containing a success message to the application.

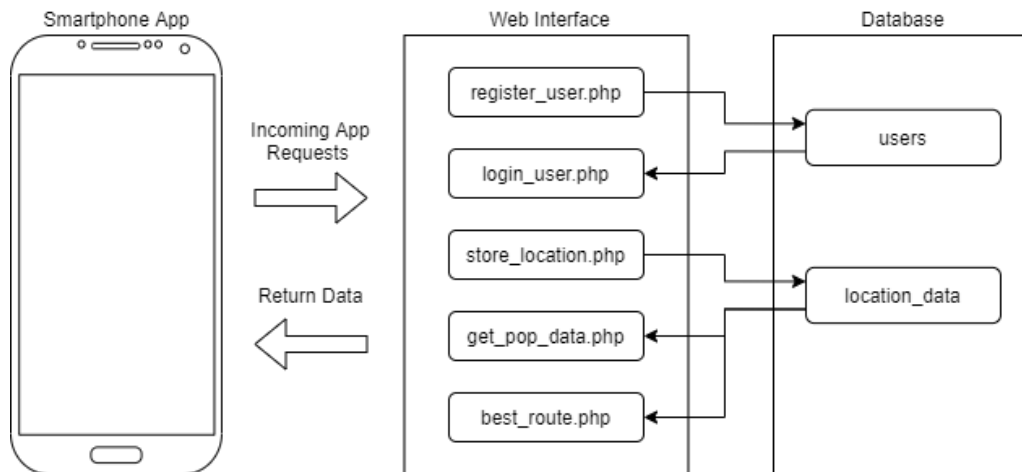


Figure 4.8: Server subsystem's interactions with the database and app.



Because the server system was designed to be as modular as possible, each of the web interface programs can interact with the database independently. Similarly, the user interactions and the location-data interactions with the database act as separate functions of the web interface and could even be hosted on different servers if necessary. This open-software development strategy was implemented to ensure that other developers could implement the framework of individual PEEPS systems without having to implement all of the PEEPS functionality.

### 4.4.3 Web Interface Structure

The web interface of the PEEPS application is hosted using the open source web server solution stack XAMPP. This application was predominantly chosen for its lightweight and cross platform compatibility, making it an appropriate method of web hosting for many situations. While XAMPP allows for a combination of different programming languages, the only one needed for this project is PHP. PHP is a general-purpose scripting language which is predominantly used for web development [25]. PHP is used in this project for data communication between the local database and the smartphone app and achieves this by manipulating JSON arrays and formatting SQL queries.

Each PHP application in this subsystem interacts similarly with the PEEPS app. This interaction within the *login\_user.php* program is as follows:

```

1  <?php
2  // Return array as JSON
3  $response = array();
4
5  //json data to array
6  $content = file_get_contents("php://input");
7  $decoded = json_decode($content, true);
8
9  //check for all required fields
10 if (isset($decoded['username']) && isset($decoded['password'])) {
11 // check if username and password is valid (not shown)
12 ...
13 } else {
14     // missing field
15     $response["success"] = 0;
16     $response["message"] = "Required field(s) is missing.";
17 //echo response

```

```

18     echo json_encode($response);
19 }
20 ?>

```

Listing 4.4: How login\_user.php interacts with the smartphone app.

This figure shows how the program interacts with the smartphone app. When the program starts, it decodes the JSON file and checks whether the correct fields are present. Should these be missing, the program fills a new array with status messages and echos them back to the app once encoded back into JSON format. Should the correct fields be present, the application will instead continue with its database query and return the outcome of the query in addition to the status messages.

### Population Retrieval Algorithm

Most of the complexity of the web interface is present in the population retrieval programs get\_pop\_data.php and best\_route.php. These programs format complicated SQL queries and return their results as one large dataset.

```

1 //connect to DB
2 $db_con = pg_connect("host=localhost port=5432 dbname=PEEPS user=postgres
   password=admin");
3
4 //assume success until failure
5 $response["success"] = 1;
6 $response["message"] = "Data received.";
7
8 for ($i=8; $i < 23; $i++) { //for each hour bracket from 8:00 to 23:00
9     $i2=$i+1; //set end time to be one hour after first
10    $count = 0;
11
12    foreach ($locations as $loc) {
13
14        //format query
15        $add_query = "SELECT COUNT(DISTINCT user_id) FROM public.location_data" .
16            " WHERE timestamp::date BETWEEN '$previous_date' AND '$current_date'" .
17            // date within 3 weeks
18            " AND timestamp::time BETWEEN time '$i:00:00' AND time '$i2:00:00'" .
19            //between hour and hour+1
20            " AND EXTRACT(DOW FROM timestamp) = " . $dow .

```

```

21     // on day of week of current date
22     " AND ST_Distance(ST_Transform('SRID=4326;POINT($loc[0] $loc[1])'
    ::geometry, 3857), ST_Transform(ST_SetSRID(coordinates,4326),3857))
    <= 20";
23     //within 20 meters of location coordinates
24
25     $result = pg_query($db_con, $add_query);
26
27     //check for errors
28     if ($result) {
29         //SUCCESS
30         $row = pg_fetch_assoc($result);
31         // $value = ($row["count"]+0)/$week_scope;
32         $response["n".strval($i)."_".strval($i2)."_loc".$count] =
            $row["count"]+0;
33
34     } else {
35         // FAILURE
36         $response["success"] = 0;
37         $response["message"] = "Data not available.";
38
39         $error = true;
40         break;
41     }
42     $count++;
43 }
44 }
45
46 //echo response
47 pg_close();
48 echo json_encode($response);
49 }
50
51 //echo response
52 pg_close();
53 echo json_encode($response);

```

Listing 4.5: get\_pop\_data.php's population-data extraction algorithm.

This algorithm queries the database for a number of unique users who have have been within 20 meters of the selected location over the last three weeks and also on the current

day of the week. It does this for each hour interval between 8 AM and 11 PM and stores this data in an array. Because only the number of users is determined, and not which users, the data is successfully anonymised. When the array is full, it will be converted to JSON format and returned to the app.

#### 4.4.4 Database Architecture

The PostgreSQL database is implemented using the program pgAdmin. This database management tool is both cross-platform and open source, and is currently the most popular platform for developing PostgreSQL databases[26]. In addition to the base PostgreSQL installation, the extension PostGIS is utilised to implement location-related objects and functions. This extension is particularly useful for determining distances between locations. This distance is utilised by the web interface to determine the number of datapoints within proximity to a saved location.

The database consists of two tables: *users* and *location\_data*:

Field Name	Primary?	Not null?	Type (size)
username	Yes	Yes	VARCHAR (15)
password	No	Yes	VARCHAR (20)

Table 4.1: ‘Users’ table field descriptions.

Field Name	Primary?	Not null?	Type (size)	Notes
entry_id	Yes	Yes	INTEGER	Autogenerated by database.
user_id	No	No	VARCHAR (15)	From ‘users’ table.
coordinates	No	Yes	GEOMETRY	Stored in (lon, lat) form.
timestamp	No	Yes	TIMESTAMP	Datatype ignores time zone.

Table 4.2: ‘Location data’ table field descriptions.

Table 4.2 opts to use the more complicated ‘geometry’ datatype over the simpler ‘point’ type which only stores x/y coordinates. PostGIS’s geometry type is built for locational data and can be manipulated to determine distances, geometrical shapes, and areas. While this project mostly uses the distance calculation functionality, the geometry datatype leaves open the possibility for far more complex locational data manipulation.

# Chapter 5

## Testing and Results

As all the necessary requirements for the PEEPS framework presented in 3.1.2 were fulfilled, testing on both the application and server systems could commence. The experimental setup and findings for this project are detailed in this section.

### 5.1 Final Product Presentation

The final product was tested on a OnePlus 7 Pro smartphone with its GPS and network location services enabled. The device contains a Snapdragon 855 processor, a 1440p display and 6GB of RAM.

The login and register GUIs are built with simplicity and usability as their core design principles. The input fields are centred and take up most of the screen's interactive space, while the actions that the user can take are displayed as highly contrasted buttons. These views also highlight the theme that was decided for this app which is a clean light grey background with purple interactable elements.

The views in Figure 5.4 and Figure 5.5 show the main functionality of the application. Figure 5.4 is the main page of the Population-Density Activity, displaying the user's saved locations and their current population status. For the purpose of testing, these locations' population data were artificially inserted into the database to provide insight into how the app would look under actual usage conditions. Figure 5.5 displays the Dialog Fragment which overlays the view of Figure 5.4. Here the user inputs the name, icon, and coordinates of a new location.

## 5.1. FINAL PRODUCT PRESENTATION

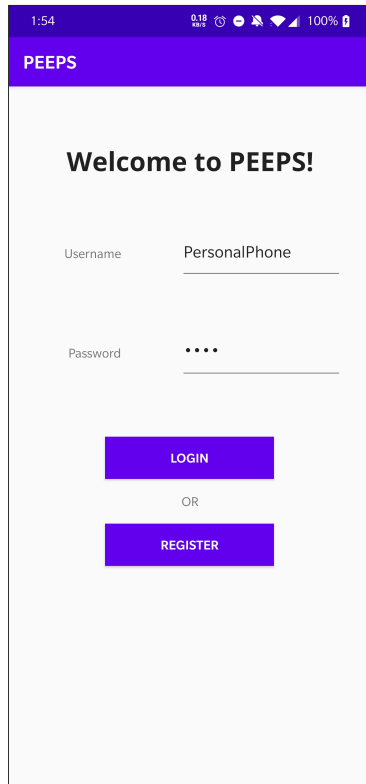


Figure 5.1: Final product login view.

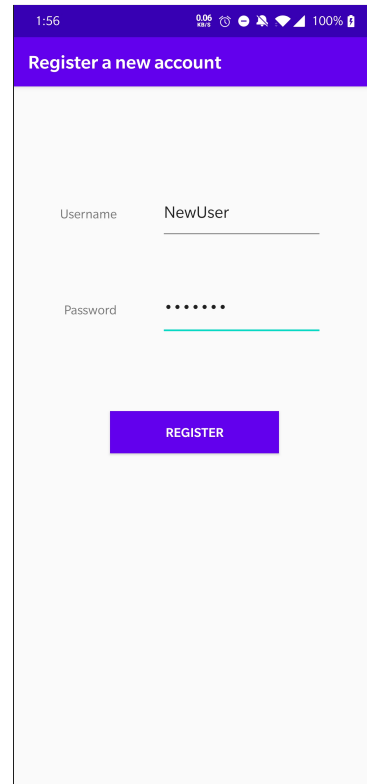


Figure 5.2: Final product register-user view.

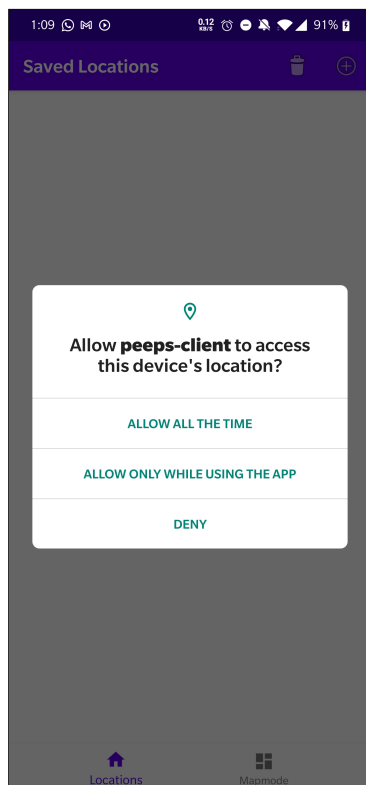


Figure 5.3: Location permission alert shown when user first logs in.

When the user first logs into the app after creating an account, they will be asked to allow the app to access their location. Without this feature, the user will not be able to contribute their location to the crowd sourcing database. The app will however still function without this feature enabled.

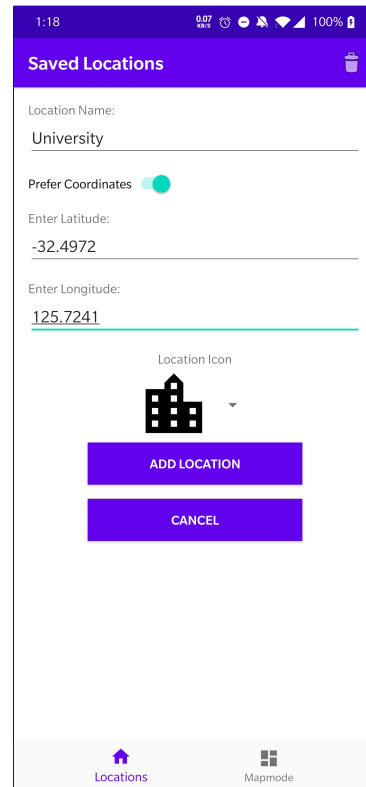
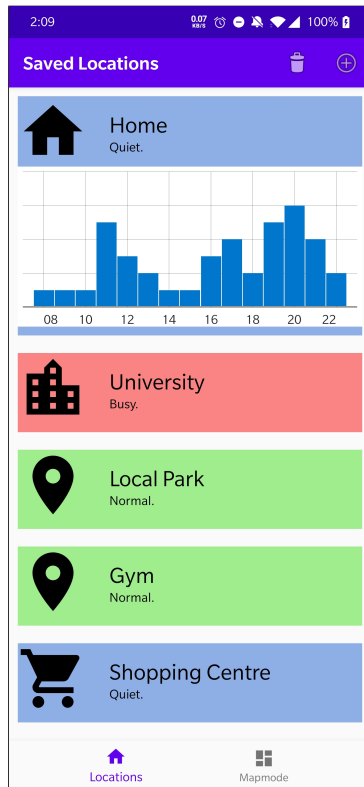


Figure 5.4: Extended saved-location view. Figure 5.5: Saved-location dialog fragment.

The GUI from the mapmode fragment is shown in the results page in Figure 5.7.

### 5.1.1 Application Interaction Feedback Mechanisms

It is important when designing a user interface that the user is presented with dynamic methods of interaction to help inform them of the results of their actions. The PEEPS application implements this by utilising ‘toasts’. These are temporary update messages which inform the user of a background process that they would benefit from knowing about.

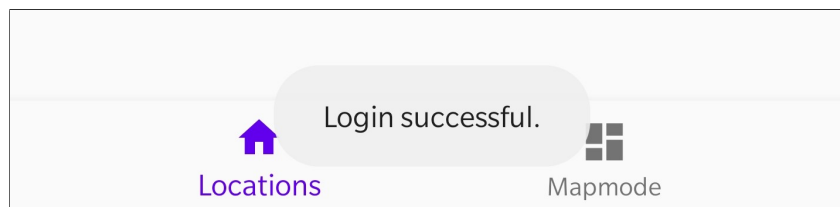


Figure 5.6: Population-Density Activity's successful login toast.

Another way the application provides feedback to the user is the transition between an item of figure 5.4 being expanded and closed. When the item is selected, the location

card will slowly expand and the graph will become visible. This transition makes the user interaction less sudden and ensures they understand the consequences of selecting an item in the list.

## 5.2 Functional Testing

### TEST-1: Location Status Test

For this first test, a list of user locations was manually added to the database with a constant set of coordinates. The number of locations added to the database was equal to the first number of the time bracket at which their location was hypothetically taken, with a timestamp dating a week earlier than the test day. Because the web interface queries data from three weeks prior to the request, we expect to see a linearly increasing graph displayed on the app when analysing data from the correct coordinates.

While conducting the test, the resulting graph displayed on the app was one with a linearly increasing population. This verifies the app's ability to use the Saved Location View to display accurate information about the status of chosen locations.

### TEST-2: Best Route Test

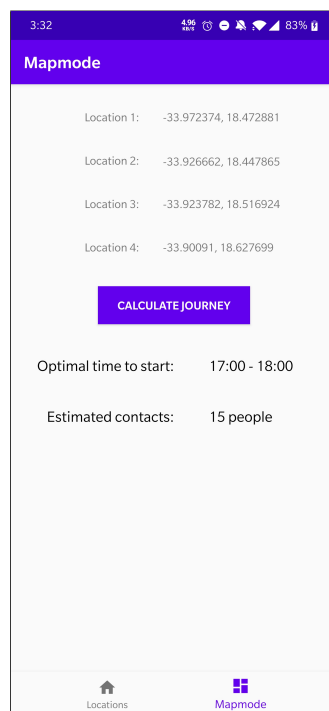


Figure 5.7: Mapmode displaying the optimal start-time and estimated contacts.



A list of user-data was manually inserted into the database for four different locations. The app's Mapmode Fragment was activated to calculate the best time to leave on a journey through each of these locations in order to minimise the number of people whom the user would be near to. The ideal departure time and amount of contact with people was output to the device. The result of the test is displayed in Figure 5.7.

The control, performed in excel, verifies that the app is able to determine the optimal time to begin the route when minimising person-to-person contact. The raw data and control can be viewed in Appendix A.1.

### **Functional Analysis**

The Location Status and Best Route tests both show that the app can function correctly while providing the user with important population information. This information can be used to make meaningful decisions like when to visit a generally crowded area, how populated a business owner's shops are and when best to go out on your planned trip to minimise the chance of infection. The success of the tests indicate that the app meets its functional requirement of providing a useful service to the user.

## **5.3 Investigative Testing**

### **TEST-3: Location Uncertainty Test**

For this test, a smartphone with the app installed was left in a stationary location for four hours so that it had sufficient time to make location updates. The normal frequency at which the location was uploaded was also artificially increased for the purpose of this test. The scatter of coordinates that were uploaded to the database was recorded and analysed. Because the distribution of the locations is assumed to be gaussian, the uncertainty was calculated using a Type A evaluation. This method uses a combination of the mean and standard deviation to determine the bounds of how accurate the location is. Note that because the smartphone's location is determined by the phone's GPS and network sensor, both the accuracy and uncertainty of location determination will differ from phone to phone.

To determine the uncertainty of the phone's location gathering capabilities, first the coordinate system was zeroed on the mean and each coordinate was converted from

degrees into the distance from the mean in meters. This was used to calculate the standard deviation and uncertainty as follows:

$$\text{standard deviation, } \sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i)^2} = 3.0525 \text{ meters}$$

$$\text{uncertainty, } u(x) = \frac{\sigma}{\sqrt{n}} = \frac{3.0525}{\sqrt{n}}$$

From this result, we can see that the uncertainty of a number of readings is dependent on the number of readings taken. For this test 25 readings were taken which provides an uncertainty of 0.6105 meters. Considering that the PEEPS framework supplies the user with data from locations within 20 meters of their chosen location, an uncertainty of this magnitude is small enough not to have any impact on the functionality of the application. However, if only one reading,  $n = 1$ , is used to estimate the user's location, the uncertainty is given to be about 3 meters. While the low uncertainty does not necessarily mean that the device's sensors are accurate, it does mean that once you have a device in a location, the coordinates it uploads will likely be within 3 meters of the mean coordinate. While this variation in location-data is acceptable for this type of crowd-sourced project, it may still be too high for applications where distances in units need to be measured.

#### TEST-4: Upload Regularity Test

Because the PEEPS framework only functions adequately if the database can receive enough user data, the regularity of the data upload is important for its ability to function reliably. The PEEPS application uses the Android Job Manager to schedule the location upload process, so its timeliness depends on how much priority the Job Manager gives to the process. For this test, the PEEPS application's Job Scheduler set the job's periodic schedule at its normal 15-minute intervals and the phone was left to upload its location over the course of four hours. The timestamps of these uploads were retrieved from the database and their mean was calculated.

This mean upload interval was determined to be 15.153 minutes. This number is only slightly off the interval set by the Job Scheduler, meaning that the Job Manager is running the service on an interval nearly identical to the one set for it. This result indicates that using a Job Scheduler was the correct method of implementing background services, as it executes reliably.

**TEST-5: Smartphone Diagnostic Test**

The app was put through a thorough test lasting 3 minutes using the full functionality of the app. The CPU, memory, network and energy usage were recorded using Android Studio’s profiler. Since the profiler does not show averages, they were calculated by sampling the recorded values.

<b>Diagnostic</b>	<b>Max</b>	<b>Min</b>	<b>Average</b>
CPU Usage (% of CPU resources)	17%	$\approx 0\%$	4%
Memory Usage	152 MB	78 MB	107 MB
Network Usage	2.4 KB/s	0 kb/s	0.1 kb/s
Energy Usage (profiler energy rating)	Light	$\approx$ none	$\approx$ none

Table 5.1: Smartphone diagnostic test results.

The application is built to be lightweight in terms of its processing requirements and this is mirrored in the results of this test. While the CPU usage does spike to 17% at its maximum, the average CPU usage of 4% means that the application will not put a large load on the phone’s capabilities. Even if the average CPU usage was closer to the maximum, since the app is running in the foreground, one would assume it would make use of most of the phone’s resources, but this is not the case. The memory usage is small enough to take up a negligible amount of the phone’s total RAM.

**Investigative Analysis**

The investigative testing implies that the PEEPS framework meets the goal of it being lightweight, scalable and sufficiently accurate. In particular, the uncertainty of the location gathering mechanism, determined by TEST-3, is a promising result for developers wishing to work with closer proximities than PEEPS’s 20-meter radius. A 0.6105-meter uncertainty, determined over the course of 25 readings, means that location uploading services could even target specific locations and determine if a person is in that area. This can be incredibly useful for businesses to determine whether an employee is at their station or for IoT devices to determine events, like whether a person has left their bed.

The strict interval of the location upload process is also promising for developers wanting a service that requires strict regularity. While the PEEPS service does not need minute precision, apps that do will find that the JobService functionality of Android applications is reliable enough for this purpose, while also having runtime customisation from network status to battery power.

While Android Studio's profiler is limited in its analysis, the usage data it provided indicates that this app is lightweight in terms of both network and smartphone resource usage. This is ideal, as the objective is to have an app that can be installed and used on any compatible device. While the test smartphone can be considered high-end, the diagnostic results indicate that the phone's processing ability is not a major concern.

## 5.4 Verification of Specifications

### Mobile Application Specifications

#### SA-1: General Specifications

SA-1.1: App must be able to run on any version of Android between 6 and 10. **Fulfilled.**

#### SA-2: Graphical User Interface

SA-2.1: Must follow a consistent colour pallet with a primary and secondary colour selection. **Fulfilled.**

SA-2.2: All user interactions with the app must be clearly interpretable from the GUI. **Fulfilled.**

SA-2.3: Buttons must be placed below text inputs in each activity or on the toolbar. **Fulfilled.**

#### SA-3: Location Services

SA-3.1: Location must be uploaded to server at regular intervals. **Fulfilled.**

SA-3.2: Location upload interval must be shorter than 20 minutes. **Fulfilled.**

SA-3.3: Location data is uploaded when app is minimised. **Fulfilled.**

SA-3.4: Location data is uploaded when app is closed. **Fulfilled.**

SA-3.5: Location data upload starts on smartphone boot if user has logged in previously. **Fulfilled.**

#### SA-4: User interaction

SA-4.1: User will be able to save 'favourite' locations by entering their coordinates or address. **Partially fulfilled. No address searching.**

SA-4.2: Saved locations will be stored locally using an SQL database. **Fulfilled.**

SA-4.3: Saved locations will display current population density status on the app's main page. **Fulfilled.**

SA-4.4: When selected, saved locations will display a graph showing the level of activity at that location for that day, as well as the predicted activity for the rest of the day. **Fulfilled.**

SA-5: Data security

SA-5.1: User will need to login with correct username and password to access application. **Fulfilled.**

SA-5.2: No population density data displayed on the app will contain information about the user who uploaded it. **Fulfilled.**

### Database Specifications

SA-1: General Specifications

SA-1.1: Database will be secured with appropriate password. **Fulfilled.**

SA-2: Tables

SA-2.1: Database will contain a table storing users' login data. **Fulfilled.**

SA-2.2: Database will contain a table storing users' geographical data. **Fulfilled.**

SA-2.3: Location table will require timestamps for each location stored as well as the username of the user who uploaded it. **Fulfilled, although username isn't checked by database.**

### Web Interface Specifications

SA-1: General Specifications

SA-1.1: Server is accessible from devices outside its home network. **Not fulfilled. Server needs to be on same home network as device.**

SA-2: Data Communication

SA-2.1: User and geographical data sent to the server must be checked for database compatibility. **Partially fulfilled.**

SA-2.2: Data sent to the server, if valid, will be stored on the connected database. **Fulfilled.**

SA-2.3: All location data processing will be done by the server, and only results will be forwarded to the application. **Partially fulfilled, much data processing still done by the app.**

SA-3: Security

SA-3.1: Data retrieved by the app from other users via the server must be anonymous.  
**Fulfilled.**

## 5.5 Discussion

Since the purpose of this project was to investigate the PEEPS's ability to correctly predict population density, the functional testing centred around trying to investigate whether this was accomplished successfully. The results of these functional tests indicate that this project goal was accomplished. Both the Saved-Location and Mapmode views provided the user with visually appealing user interfaces which the user could use to get actionable insight into the population density of chosen locations. The Saved-Location view was particularly successful as it allowed the user fine control over the locations they wished to analyse while also providing the predicted population for most of the day, based on past population densities. While the Mapmode view worked as intended, it acted as more of a test of the app's capabilities when dealing with large amounts of location data and processing time. The results from TEST-2 indicate that this fragment was able to correctly calculate the optimal time to begin a journey and the number of predicted contacts. Since the functionality of this view is proven, it could now be redesigned with user interaction in mind.

The investigative testing revealed useful information about both app development and Android's location services. The Fused Location Provider used by the app to find its current location was found to have a standard deviation of 3.0525 meters which is a value that is vital to know when designing location-determining apps. Any Android application using this fused-location service should be aware that the coordinate value determined will vary about two standard deviations<sup>1</sup> from the mean. The standard deviation value is very similar to the average uncertainty calculated by the Warnell School of Forestry and Natural Resources using the iPhone 6, which found that most values varied between 3 meters of the mean error amount [18]. The investigative testing also determined the reliability of the Android's background Job Service Scheduler and the lightweight execution of the PEEPS app. The latter is promising for developers wanting to use location tools in the background of a resource intensive app, as the location services used in this project were not CPU, network, memory, or battery intensive.

---

<sup>1</sup>For 95.4% of values.

# Chapter 6

## Conclusions

With COVID-19 bringing population prediction into the public consciousness, the need for an understanding of how this could be implemented was manifest. The success of this project, as presented in its brief, hinged on its ability to derive meaningful information about the process of app development, and its ability to gather insight into smartphones' capability to act as crowd-sourced location data sources. To begin the project's investigation, the literature surrounding the project objective was analysed, with existing services used as reference for how the project should approach its objective. This research assisted the formulation of the project's strategic approach, objectives, requirements, and specifications which would be used to guide the development process.

The design phase commenced with a discussion about the possible approaches towards important project design concepts. After the chosen approach was decided upon, the systems constituting the PEEPS framework were designed. The implementation phase then began with the creation of a dedicated database and web interface. Once the web interface was able to manipulate the database's data correctly, development on the PEEPS app was started. The completion of the app marked the beginning of this project's testing phase.

The results gathered from the PEEPS application's functional and investigative testing are promising for developers wishing to implement crowd-sourced location functionality, using the PEEPS framework, as the app was able to provide the user with meaningful, actionable, and relevant data that was both relatively certain and timely. The functional testing verified that the app was able to gather useful data from the database and present it to the user.

All three investigative tests provided useful insights into the operation of this device. The location uncertainty test, Test-3, determined that there is a standard deviation of 3.0525 meters from the mean when uploading a device's location. Since 68% of recorded values are calculated to be within one standard deviation of the mean, we can expect an uncertainty of less than three meters for most readings. Similarly, we can expect an variation of 6.1050 meters for 95% of recordings. However, these values can be decreased by increasing the number of recordings taken, at the cost of processing time.

Test-4, which tested the consistency of Android's job manager, found that the app's location upload JobService ran almost exactly as specified. This is promising for developers wanting an upload schedule with a consistent upload interval within seconds of what the program specified. Test-5 also displayed the app's reliability determining that the app's resource usage was minimal and should run on any compatible device.

Consequent to the testing, the list of specifications was verified, and a discussion was conducted about whether the project's implementation was successful. Finally, conclusions were drawn, and recommendations were given based on the end-product delivered by the project.



# Chapter 7

## Recommendations and Future Development

### 7.1 Recommendations

#### 7.1.1 Flutter as a Substitute for Java

While this project used Android as its method of implementing the app, it is not recommended for developers who wish to develop for both Android and iOS. With Flutter's surge in popularity, it is now the current best solution to multi-OS development. Because some background processes, when developing using Flutter, still need dedicated Java code, developers will be able to make use of the PEEPS app's background processing codebase as a template.

#### 7.1.2 Fullscreen Dialog-Fragment

Since Fullscreen Dialog-Fragments are Google's recommended method of large data input from an overlay, it was chosen as the method for inputting the user's saved location details<sup>1</sup>. However, the shortcomings of this input method become apparent as soon as it was implemented. Not only does it negatively affect performance while using the app, but Google's own documentation of this type of input is sparse. Thus, it is recommended

---

<sup>1</sup>As displayed in Figure 4.5.

that developers use dedicated views instead of dialog fragments when possible for complex inputs containing many values or custom input mechanisms.

### 7.1.3 Data Processing

In the current version of PEEPS, most of the data processing is handled by the server in the form of SQL formatting and execution. While the performance testing showed that the app was not resource intensive while still doing a large amount of raw data processing, this may not be the case for older generation devices. Limiting the amount of data sent to the device also prevents hackers from gaining access to sensitive information. Hence, it is recommended to perform further data analysis on the server and relay only the information which the app needs to display.

## 7.2 Future Development

### 7.2.1 Extensive Network Testing

Network diagnosis was not heavily focussed on in the investigative testing section of this report. This is largely due to Android's lack of diagnostic tools able to measure this value. While the Android Studio Profiler shows the usage over time, it does not report values like total network usage over time. In conducting further testing, one could create a Java class that measures the network usage that can determine whether the program is lightweight enough for users to want to use it. This is important for the South African use-case, as data is far less affordable than in other countries. Such a class could additionally be used to gather more accurate diagnostic information about the app's operation.

### 7.2.2 Population-Scale Testing

Due to the novel coronavirus epidemic, large scale population testing was not in the scope of this project. Such a test would involve a large number of individuals downloading the app on their phones so that the database would have enough data to simulate a commercial use-case. The results of this test would provide detailed information about the real-world applications of the PEEPS software, such the app's usability, effectiveness,

and applicability to users.

### 7.2.3 iOS Comparison Testing

While Android is more popular worldwide, the US market is dominated by Apples smartphones. This means that if one wants to enter their large Appstore marketplace, developing for iOS devices becomes essential. Therefore, future development could replicate the PEEPS app using Swift or Flutter so that multi-OS testing can be conducted, and differences in these operating systems can be investigated.

# Bibliography

- [1] M. Priyadarshini, “Which Sensors Do I Have In My Smartphone? How Do They Work?” *Fossbytes*, sep 2018. [Online]. Available: <https://fossbytes.com/which-smartphone-sensors-how-work/>
- [2] T. Bush, “5 Powerful Alternatives to Google Maps API,” *Nordic APIS*, nov 2018. [Online]. Available: <https://nordicapis.com/5-powerful-alternatives-to-google-maps-api/>
- [3] F. Carozzi, “Urban Density and Covid-19,” *Social Science Research Network*, vol. IZA Discus, no. No. 13440, p. 29, 2020. [Online]. Available: [https://papers.ssrn.com/sol3/papers.cfm?abstract{\\_  
}id=3643204{#}references-widget](https://papers.ssrn.com/sol3/papers.cfm?abstract={_}id=3643204{#}references-widget)
- [4] ICASA, “The State of the ICT Sector Report in South Africa,” Independent Communications Authority of South Africa, Tech. Rep., 2020. [Online]. Available: <https://www.icasa.org.za/uploads/files/State-of-the-ICT-Sector-Report-March-2020.pdf>
- [5] A. Turner, “HOW MANY SMARTPHONES ARE IN THE WORLD?” 2020. [Online]. Available: <https://www.bankmycell.com/blog/how-many-phones-are-in-the-world> [Accessed: 2020-10-08]
- [6] Allied Market Research, “Mobile Application Market to Garner \$311.25 Billion, Globally, By 2023 at 19.2% CAGR, Says Allied Market Research,” 2019. [Online]. Available: <https://www.prnewswire.com/news-releases/mobile-application-market-to-garner-311-25-billion-globally-by-2023-at-19-2-cagr-says-allied-mar.html> [Accessed: 2020-10-07]
- [7] 42 matters, “South Africa App Market Statistics in 2020 for Android.” [Online]. Available: <https://42matters.com/south-africa-app-market-statistics> [Accessed: 2020-11-03]
- [8] Cable.co.uk, “Worldwide mobile data pricing: The cost of 1GB of mobile data in 228 countries.” [Online]. Available: <https://www.cable.co.uk/mobiles/worldwide-data-pricing/> [Accessed: 2020-11-03]

- [9] E.-J. Bottomley, “SA has some of Africa’s most expensive data, a new report says – but it is better for the richer,” may 2020. [Online]. Available: <https://www.businessinsider.co.za/how-sas-data-prices-compare-with-the-rest-of-the-world-2020-5>
- [10] S. Stein, “Lidar on the iPhone 12 Pro: What it can do now, and why it matters for the future of AR, 3D scanning and photos,” *CNET*, oct 2020. [Online]. Available: <https://www.cnet.com/how-to/lidar-on-the-iphone-12-pro-what-it-is-and-why-it-matters-for-the-future-of-ar-3d-scanning-and-pho>
- [11] L. Barkhuus and A. Dey, “Location-Based Services for Mobile Telephony: a Study of Users’ Privacy Concerns.” vol. 2003, 2003.
- [12] P. Jäppinen, I. Laakkonen, V. Latva, A. Hämäläinen, and J. Porras, “Bluetooth Device Surveillance and Its Implications,” 2004.
- [13] K. Geyer, D. A. Ellis, and L. Piwek, “A simple location-tracking app for psychological research,” *Behavior Research Methods*, vol. 51, no. 6, pp. 2840–2846, 2019. [Online]. Available: <https://doi.org/10.3758/s13428-018-1164-y>
- [14] M. J. Chorley, R. M. Whitaker, and S. M. Allen, “Personality and location-based social networks.” *Computers in Human Behavior*, vol. 46, pp. 45–56, 2015.
- [15] S. Saeb, E. G. Lattie, S. M. Schueller, K. P. Kording, and D. C. Mohr, “The relationship between mobile phone location sensor data and depressive symptom severity.” *PeerJ*, vol. 4, p. e2537, 2016.
- [16] D. Williams II, “The History of Augmented Reality (Infographic),” 2017. [Online]. Available: [https://www.huffpost.com/entry/the-history-of-augmented-{}\\_b{}\\_9955048](https://www.huffpost.com/entry/the-history-of-augmented-{}_b{}_9955048) [Accessed: 08/11/2020]
- [17] BuiltWith, “Google API Usage Statistics.” [Online]. Available: <https://trends.builtwith.com/javascript/Google-API> [Accessed: 2020-11-03]
- [18] K. Merry and P. Bettinger, “Smartphone GPS accuracy study in an urban environment,” *PLOS ONE*, vol. 14, no. 7, p. e0219890, jul 2019. [Online]. Available: <https://doi.org/10.1371/journal.pone.0219890>
- [19] Merriam-Webster, “Crowdsourcing.” [Online]. Available: <https://www.merriam-webster.com/dictionary/crowdsourcing> [Accessed: 2020-10-10]
- [20] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, “The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring,” in *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008)*, Breckenridge, U.S.A., jun 2008.

- [21] A. Yerukhimovich, R. Balebako, A. E. Boustead, R. K. Cunningham, W. W. IV, R. Housley, R. Shay, C. Spensky, K. D. Stanley, J. Stewart, A. Trachtenberg, and Z. Winkelman, *Can Smartphones and Privacy Coexist? Assessing Technologies and Regulations Protecting Personal Data on Android and iOS Devices*. Santa Monica, CA: RAND Corporation, 2016.
- [22] M. Anderson, “More Americans using smartphones for getting directions, streaming TV,” jan 2016. [Online]. Available: <https://www.pewresearch.org/fact-tank/2016/01/29/us-smartphone-use/>
- [23] Associated Press, “Google records your location even when you tell it not to,” aug 2018. [Online]. Available: <https://www.theguardian.com/technology/2018/aug/13/google-location-tracking-android-iphone-mobile>
- [24] StatCounter, “Mobile Operating System Market Share Worldwide.” [Online]. Available: <https://gs.statcounter.com/os-market-share/mobile/worldwide> [Accessed: 2020-01-15]
- [25] The PHP Group, “PHP.” [Online]. Available: <https://www.php.net/> [Accessed: 2020-11-03]
- [26] PgAdmin, “pgAdmin Frontpage.” [Online]. Available: <https://www.pgadmin.org/> [Accessed: 2020-11-03]

# Appendix A

## Additional Files

### A.1 TEST-2 Data and Control

L	8:30	9:30	10:30	11:30	12:30	13:30	14:30	15:30	16:30	17:30	18:30	19:30	20:30	21:30	22:30	23:30
loc 0	8	7	5	9	12	15	10	6	8	7	9	16	17	15	12	6
loc 1	2	4	5	2	3	6	7	4	1	2	1	5	3	6	2	2
loc 2	4	5	3	2	3	0	1	4	7	7	9	7	7	8	10	6
loc 3	4	5	4	3	4	5	8	4	6	9	4	3	2	0	5	8

Table A.1: Input data for TEST-2.

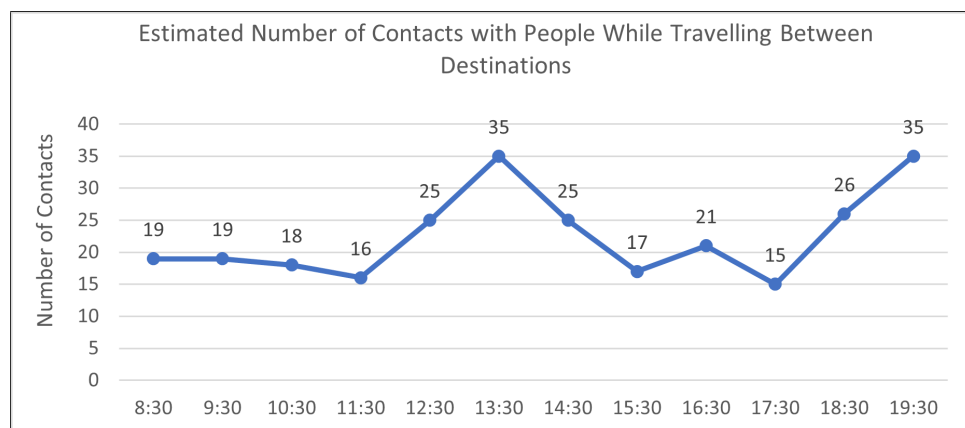


Figure A.1: Control for number of contacts in TEST-2, created using excel.

# Appendix B

## Addenda

### B.1 Github link

The app, server and report software are accessible from [github.com/ianpsgrantZA/PEEPS](https://github.com/ianpsgrantZA/PEEPS)

### B.2 Ethics Form



## ETHICS APPLICATION FORM


**Please Note:**

Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: <http://www.ebe.uct.ac.za/ebe/research/ethics1>

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Ian Grant
Department		Electrical Engineering
Preferred email address of applicant:		grnian004@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	BScEng in Elec & Computer Eng
	Credit Value of Research: e.g., 60/120/180/360 etc.	40
	Name of Supervisor (if supervised):	Simon Winberg
If this is a research contract, indicate the source of funding/sponsorship		n/a
Project Title		PEEPS – Population of Environment Evaluation and Prediction System

**I hereby undertake to carry out my research in such a way that:**

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
<b>Principal Researcher/ Student/External applicant</b>	Ian Grant		17/08/2020
SUPPORTED BY	Full name	Signature	Date
<b>Supervisor (where applicable)</b>			

APPROVED BY	Full name	Signature	Date
<b>HOD (or delegated nominee)</b> Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).			
<b>Chair: Faculty EIR Committee</b> For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			