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 4, 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.

Signature:...........................
M. S. Tšoeu

Date:..............................

Acknowledgments

Acknowledgements go here.

Abstract

- Open the **Project Report Template.tex** file and carefully follow the comments (starting with %).
- Process the file with **pdflatex**, using other processors may need you to change some features such as graphics types.
- Note the files included in the **Project Report Template.tex** (with the .tex extension excluded). You can open these files separately and modify their contents or create new ones.
- Contact the latex namual for more features in your document such as equations, subfigures, footnotes, subscripts & superscripts, special characters etc.
- I recommend using the **kile** latex IDE, as it is simple to use.

Contents

List of Figures

List of Tables

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 in locations with high population density, it is in the interest of the public's health to minimise excess 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 lines as it is for a new business owner to know what is causing the traffic levels in their store. Hence, both population evaluation and population prediction have the potential to revolutionise how we predict prosperity and enhance our own lifestyle.

Large scale population data tracking has become increasingly viable in the last decade due to the precise positioning sensors on smartphones and their extensive prevalence. 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 Google's 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-focussed design to allow for future developers to choose which aspects of population evaluation they wish to use. 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 on non-research individuals be conducted. Since the PEEPS system's objective is for it to be used as a framework, this creates a difficulty for it to become a publish-ready application, as the program's focus will be on creating systems that can be incorporated by other projects, rather than being a completely fleshed out application.

1.4 Plan of development

The project was initiated by performing a thorough analysis on the landscape of app development and population prediction mechanisms. This review was then 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 at a functional standard, development on the smartphone app begun. 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 Framework, 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.

1.5 Plan of development

TODO

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 in even 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 [3]. This combined with near total 3G network coverage [3] 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 more closely understand what advancements the smartphone revolution has in store for us.

2.1.1 App Development

Android and iOS app development have never been easier with the release of 1st 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 [4], has made smartphone application development a priority for companies wanting access to this $311.25 billion market [5].

South Africa is no exception to this economic surge with over four thousand apps available on the Google Play Store with nearly 200 million downloads across them [6]. 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, which may not seem that prohibitive when compared to the United States (188th) and New Zealand (180th) [7], but doesn't take into account the large proportion of the population that 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 are 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 [8]. 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 [9]. The sheer number of sensors in smartphones provide 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 highly accurate approximation of the user's current position.

2.2 Data Collection

Our collective overreliance on the use of smartphones have 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 amount 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.

2.2.1 Crowd Sourced Data

The universal nature 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 an 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” [10]. This definition indicates that there are a few key criteria that a smartphone App needs to meet for it to be implementing crowdsourcing. Firstly, it must be collecting data that is being used to supply 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 userbase 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 crowdsourcing 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, where 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 phones built in vibration sensors to track and report pothole locations [11]. 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.

2.2.2 Location Tracking Services

There are two primary mechanisms of determining your location via mobile connection: position-tracking services and position-aware services. Location-tracking services entail using 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 [12]. 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 device 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.

Phones themselves have plenty of tools at their disposal capable of determining their location, from highly accurate location services like GPS, GSM and Wi-Fi positioning systems to more subtle methods like time-zone and humidity sensing. 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 [13]. 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-aware services, it becomes a cost sink that will keep expanding with the userbase.

However there are many location aware alternatives to the Google's location-tracking behemoth, but 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

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].

functionality provides developers with an innovative tool to try new ideas while allowing them a plenitude of free requests daily.

2.2.3 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.

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 3rd party server solutions and their features.

The solutions presented in Table 2.3 are very similar in their functionality so the choice between them 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.

2.2.4 Smartphone User Privacy

The number of sensors present in smartphones, as shown in table 1, means that this small handheld device gathers a disproportioned 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 apps' processes starting after a device has booted [14]. 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[cite]. 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[14]. 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 app's 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 user's and found that 90% of smartphone users use apps that rely on locational data, up from 74% in 2013 [15]. 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 [16].

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 used 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 Framework, 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 was decided to be 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 reiteration implemented while using this strategy, the Waterfall Model performs best under circumstances where the project's requirements are well documented. This is the case with PEEPS as the goal is to produce a product that can then undergo thorough testing. Any shortcoming of the app discovered by the testing phase are not design failures but instead insights into how the app could be implemented better. Likewise, the successes of the app act as useful observations for others wishing to implement similar functionality into their own app.

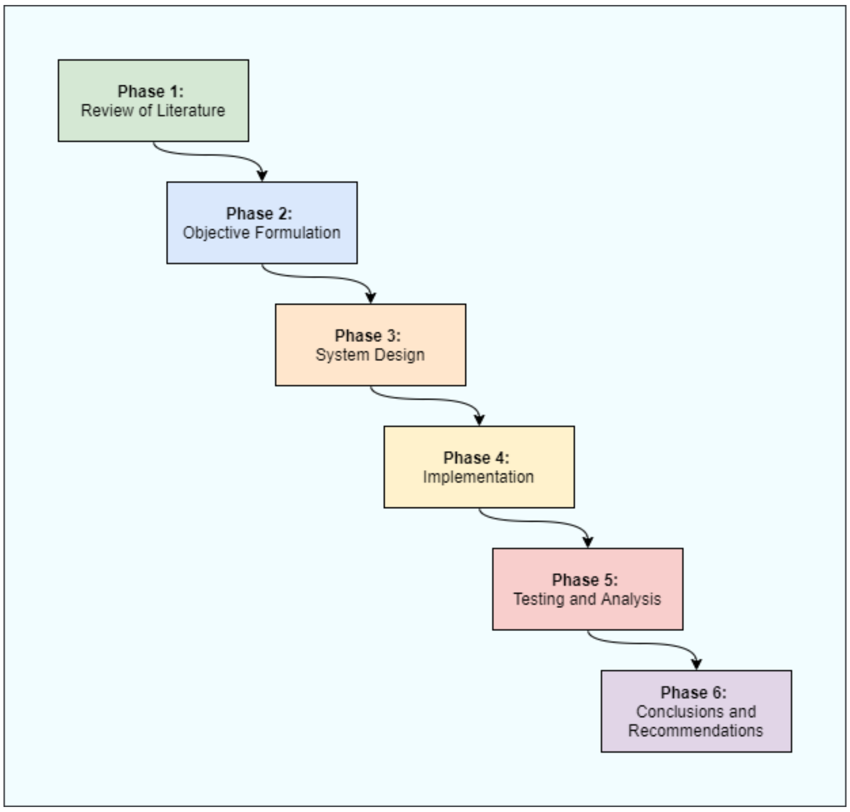


Figure 3.1: Project phase breakdown

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 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 projects development.

3.3 Project Requirements

The project requirements are the minimum nessesary criterion for which the deliverable can be described as a successful implementation. Likewise, completion of these requirements are nessesary 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 users' login details,
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 availability to researchers.
- An internet connection will be required for the application to perform.

3.5 Technical Specifications

These technical specifications detail the more specific aspects of the deliverable and, while not nessesary 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 in 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.
- 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 and 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

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 by using these proposals, justify the final design of the app and server components. A system level view will then be discussed 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.

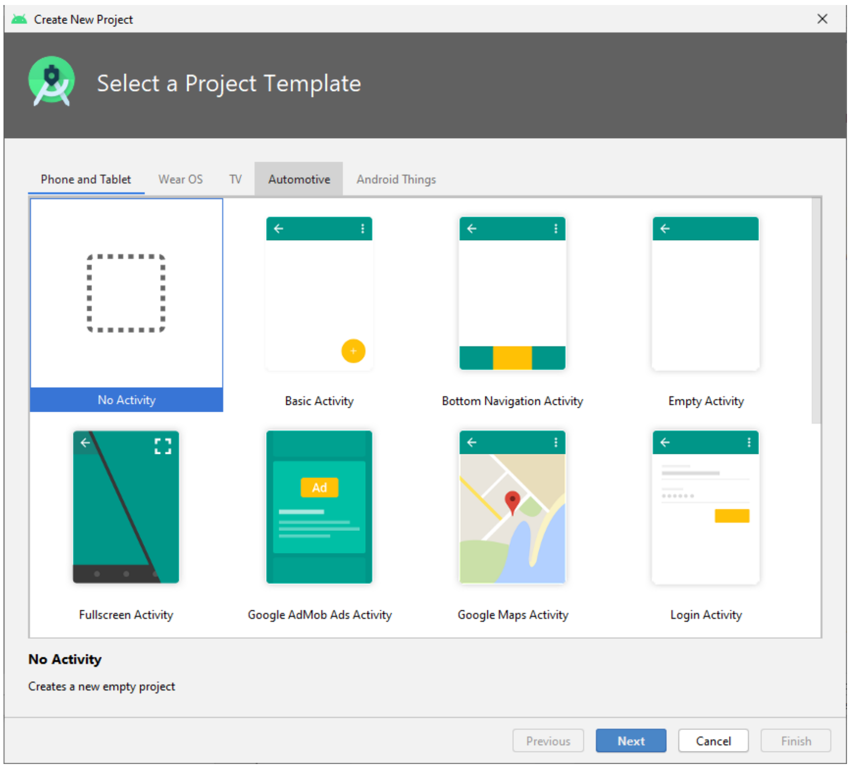
4.1.1 Concept Design Proposals

A. Smartphone Application Opperating 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 [17] and applications for each of these operating system 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, the decision of which to develop for needs to be 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 application programmed using Java:** Android devices make up over 74% of global mobile smartphones, making android development the most beneficial in terms of potential reach [17]. 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.



No Activity

Creates a new empty project

Figure 4.1: Android studio comes with activity templates 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 Apples

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 means 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 application 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 with the database, their 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 beyond just web development. Additionally, Python as a language has an elegance to is syntax that is not mirrored in PHP, making its far more interpretable when being worked on by more than one developer., but this is offset by the sheer robustness of MySQLs database implementation.
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[cite]. 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 far more popular than PHP for general use applications, PHP is still far more popular when it comes to web applications[cite].

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 3.2.1. 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 predominantly chosen due to its far 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

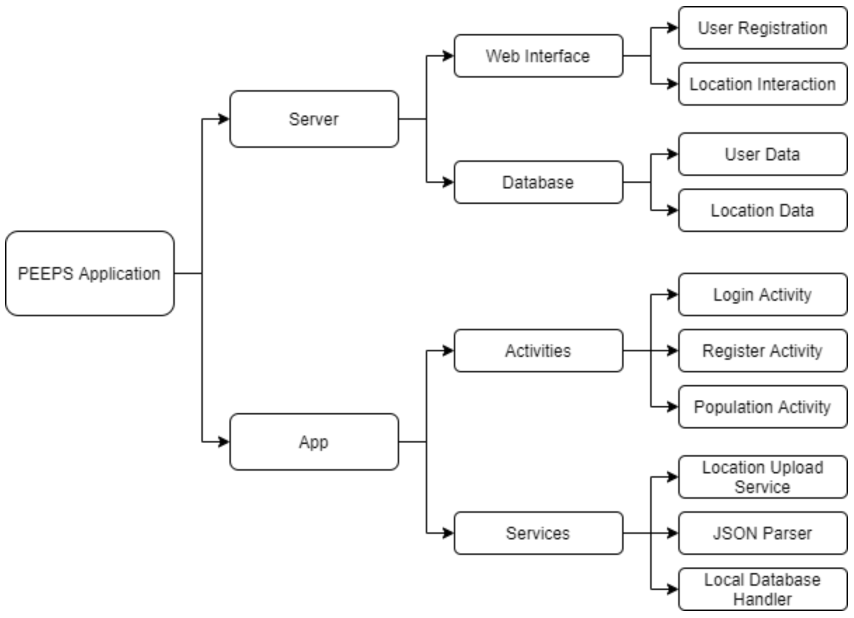


Figure 4.2: High Level Overview of main application systems and their subsystems

The design of the PEEPS framework is split into two independent systems: the smartphone application and the server host. 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 instead or the server could receive data from a different application and the fidelity of the overall framework remains operational. The inter-system interactions will be summarised in section 3.3.2 and will be discussed in more detail in the implementation section[link]. TODO

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 store or retrieve geographical data. When this is not happening, the server will act 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 up-to-date data be displayed, malicious users could abuse the system and use live population numbers 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 application. They allow the user to create an account and access the app's main functionality by logging into the app using it. 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 applications functionality, which is split into the 'saved location' and ‘mapmode’ fragments. These will be discussed in more detail in 3.4.2. TODO

4.3.2 Java Class Structure

TODO: java structure

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 both entire activity UIs and individual components of an activities 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 start 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 opperation of these services are expanded upon in Section 4.3.3.

Adapters

Adapters are used by classes which 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 in 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. Bellow is an example of the register subsystem sending user data to the server:

TODO add code

When the JSON Parser's makeHttpRequest method is called, 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 Googles battery-efficient fused location API which used a combination of the various receivers on the phone to pinpoint the phone's location.

After the fused location client has finished requesting a location update, a new thread will be created 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.

4.3.4 SQLiteDBHelper

Of the three services in the application, the SQLiteDBHelper is the only one that runs on the main thread. Its job is to set up and monitor the app's local SQLite saved-location database. This database is accessed by creating a cursor that can traverse the entries that are selected from the query:

TODO add code

Entries are inserted using the following method:

TODO add code

4.3.5 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 users' 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 crowdsourcing 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:

TODO add code

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 adequate for the execution of the PEEPS application as a balance needs to be kept between maximising useful data and minimising network data usage.

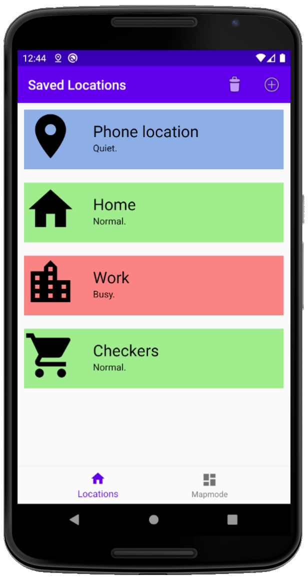
4.3.6 Population Prediction Functionality

The Population-Density Activity provideds 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 explain 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 4.3 is what the app will display once the user has added any 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



Phone location
Quiet.

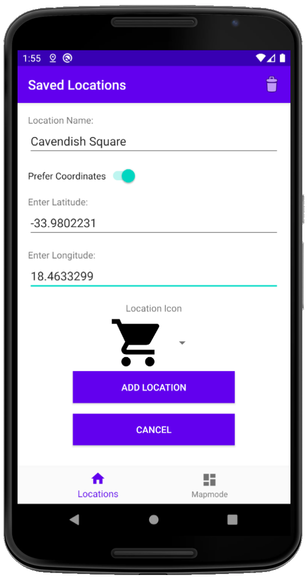
Home
Normal.

Work
Busy.

Checkers
Normal.

Figure 4.3: 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 4.4.



Location Name:
Cavendish Square

Prefer Coordinates

Enter Latitude:
-33.9802231

Enter Longitude:
18.4633299

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

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 4.5.

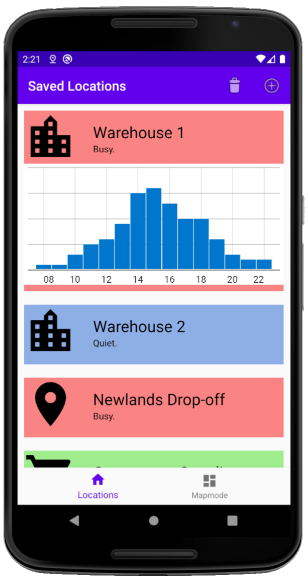
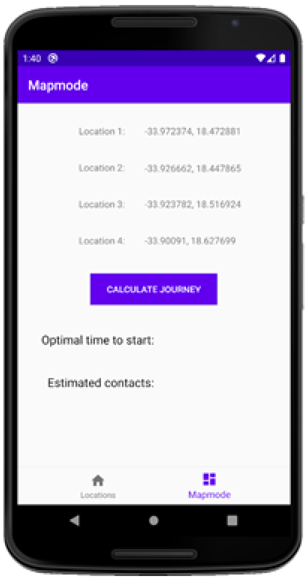


Figure 4.5: Location Fragment displaying a user's saved locations with their population status

Figure 4.5 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 at each time throughout the day. Each bar in the graph is an hour interval which displays the number of unique users who uploaded their location within 20 meters of the location's coordinates. More information on what data is shown can be found in 3.5.3. Since the time in figure 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 quite 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.



Location 1: -33.972374, 18.472881

Location 2: -33.926662, 18.447865

Location 3: -33.923782, 18.516924

Location 4: -33.90091, 18.627699

Optimal time to start:

Estimated contacts:

Figure 4.6: Mapmode Fragment's view displaying four pre-selected locations.

When the user selects this view, they will be presented with four predetermined locations and an interactable button. The coordinates of these locations are set up within the app 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 car 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 3rd 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 query the database to determine if there are conflicting usernames. 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.

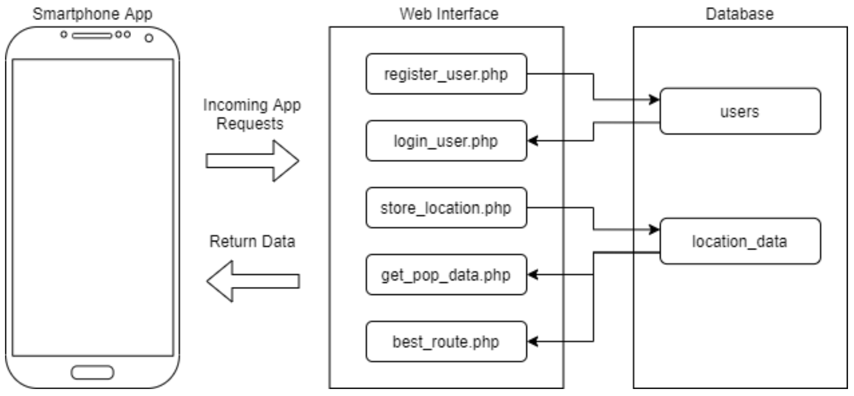


Figure 4.7: Mapmode Fragment's view displaying four pre-selected locations.

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 functions 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 [18]. 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:

TODO add code

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.

TODO explain and add code

4.4.4 Database Architecture

The PostgreSQL database is implemented using the program pgAdmin. This database management tools is both cross-platform and open source, and is currently the most popular platform for developing PostgreSQL databases[19]. 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 which is utilised to determine the number of datapoints within proximity to a saved location.

The database consists of two tables: *users* and *location_data*:

TODO add tables

This first table is a simple implementation of a user database with unique usernames.

TODO same

This table 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

Results

As all the necessary requirements for the PEEPS framework presented in 3.1.2 were fulfilled, testing on the both the application and server systems could commence. The experimental setup and findings for this project are details 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 its 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 X is the main page of the app displaying the user's saved locations and their current population status. For the purpose of testing these locations' population data was artificially inserted into the database to provide insight into ow 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.

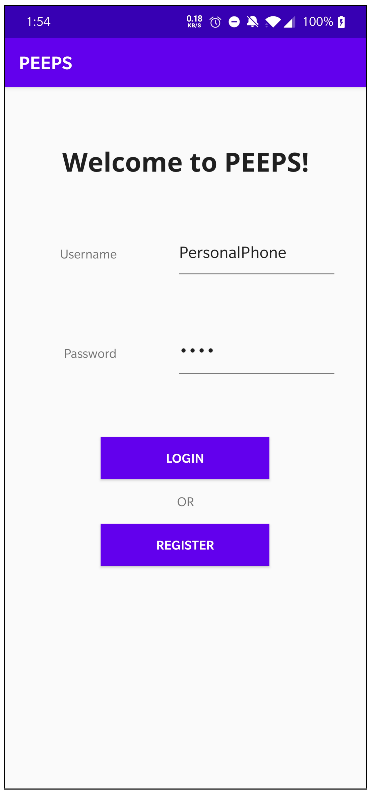


Figure 5.1: Login-and-welcome view.

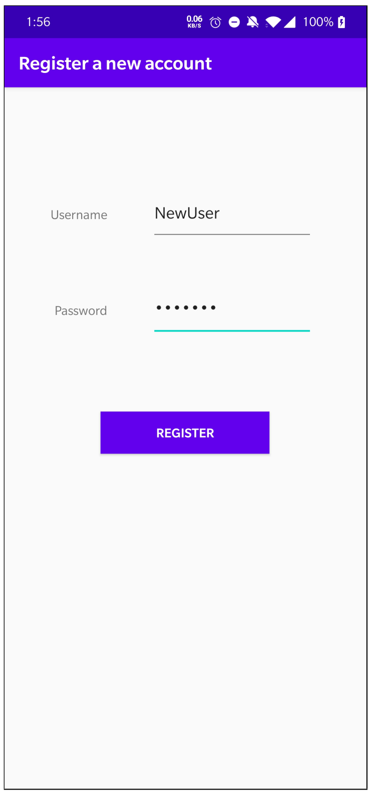


Figure 5.2: Register-user view.

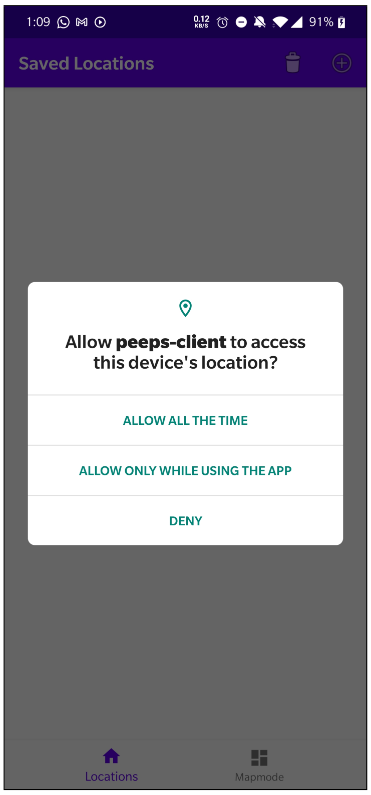
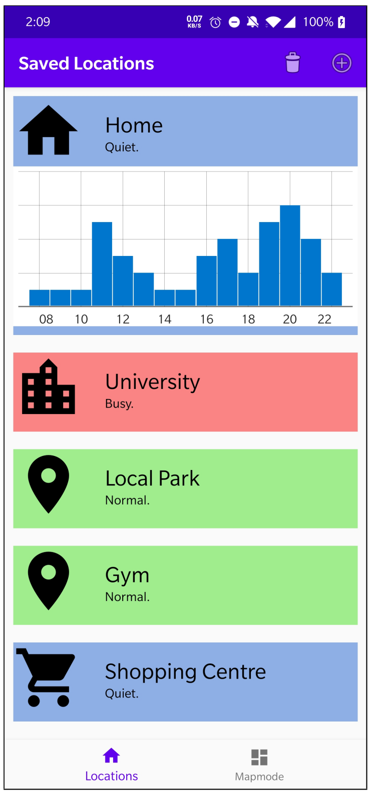

1:09
91%
Saved Locations
Allow peeps-client to access this device's location?
ALLOW ALL THE TIME
ALLOW ONLY WHILE USING THE APP
DENY
Locations
Mapmode

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

When the user first logs into the app after making 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.



Home
Quiet.

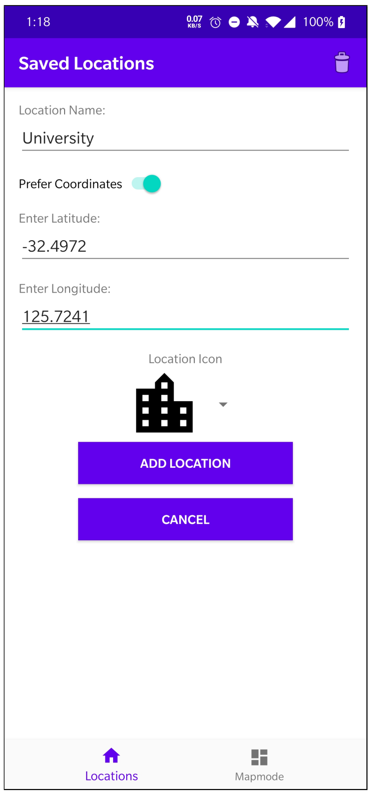
University
Busy.

Local Park
Normal.

Gym
Normal.

Shopping Centre
Quiet.

Figure 5.4: Extended saved-location view.



Location Name: University

Prefer Coordinates: on

Enter Latitude: -32.4972

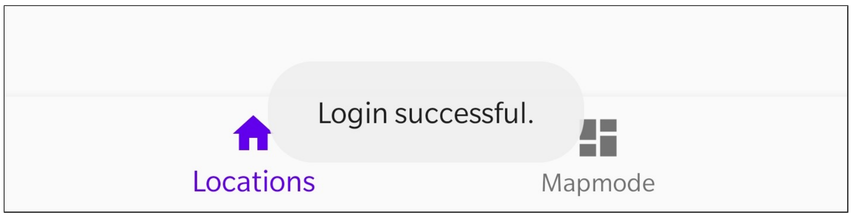
Enter Longitude: 125.7241

Figure 5.5: New-saved-location view with 'prefer coordinates' selected.

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

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 displaying 'toasts'. These are temporary update messages which inform the user of a background process that they would benefit from knowing is happening.



Login successful.

Figure 5.6: 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 amount of locations added to the database were equal to three times 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 averages data from the previous three weeks before the request, we expect to see a linearly increasing graph on the app when analysing data from the correct coordinates. When the same coordinates were added as a saved location to the app, the resulting graph displayed 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**

A list of user's 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 to minimise the amount of people who the user would be in close proximity to. The ideal departure time and amount of contact with people was output to the device. The data added can be viewed in appendix TODO.

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 control can be viewd in apendix TODO.

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 owners' shops are and when best to go out on your planned trip to minimise 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

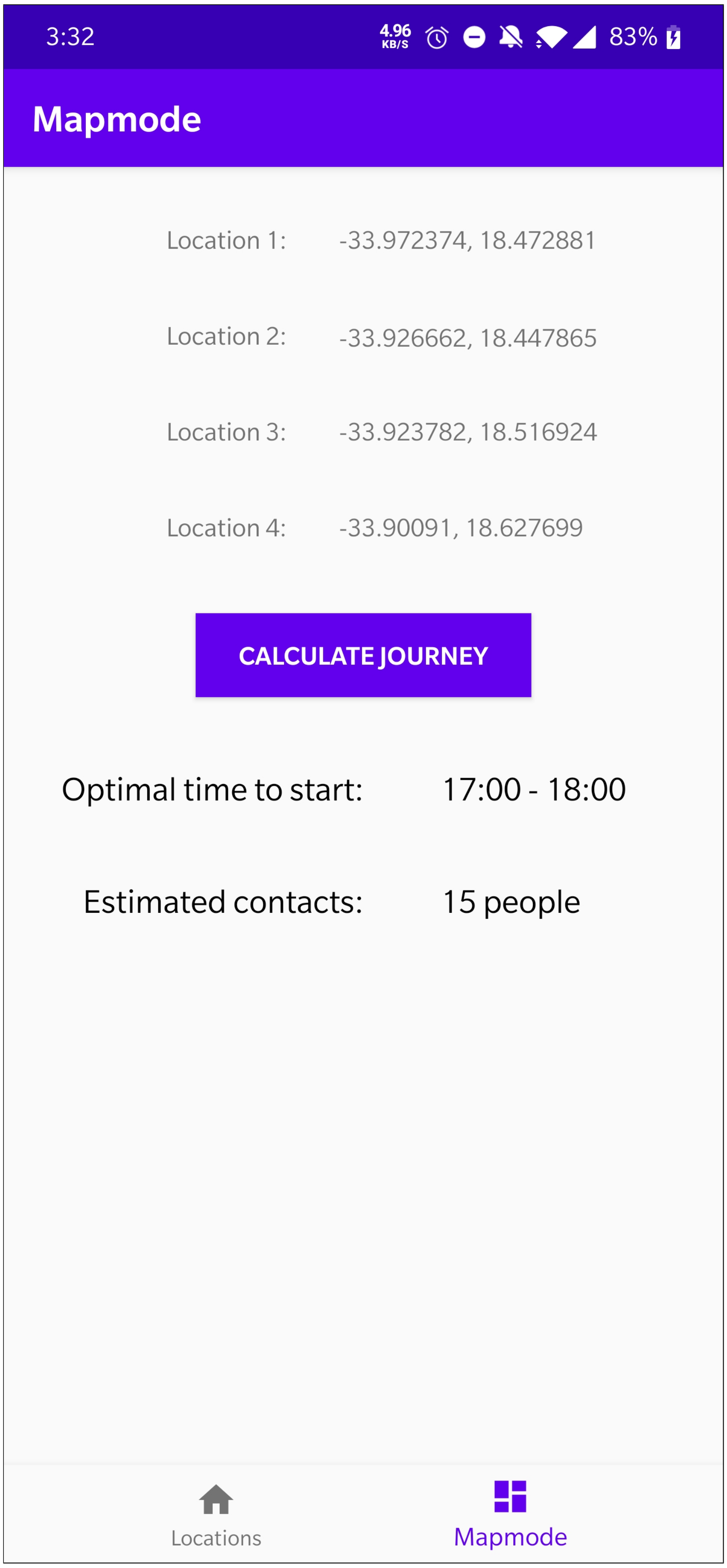
TEST-3: **Location Accuracy Test**

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 were 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, the accuracy of location gathering will differ from phone to phone.

To determine the uncertainty of the phones 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:

5.4 Verification of Specifications

5.5 Discussion



Location 1:	-33.972374, 18.472881
Location 2:	-33.926662, 18.447865
Location 3:	-33.923782, 18.516924
Location 4:	-33.90091, 18.627699

Optimal time to start:	17:00 - 18:00
Estimated contacts:	15 people

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

Chapter 6

Discussion

Here is what the results mean and how they tie to existing literature...

Discuss the relevance of your results and how they fit into the theoretical work you described in your literature review.

Chapter 7

Conclusions

These are the conclusions from the investivation and how the investigation changes things in this field or contributes to current knowledge...

Draw suitable and intelligent conclusions from your results and subsequent discussion.

Chapter 8

Recommendations

Make sensible recommendations for further work.

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] 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

[4] 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]

[5] 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: 07/10/2020]

[6] . matters, "South Africa App Market Statistics in 2020 for Android." [Online]. Available: https://42matters.com/south-africa-app-market-statistics [Accessed: 2020-11-03]

[7] 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]

[8] 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

[9] 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-pl

[10] Merriam-Webster, "Crowdsourcing." [Online]. Available: https://www.merriam-webster.com/dictionary/crowdsourcing [Accessed: 2020-10-10]

[11] 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.

[12] L. Barkhuus and A. Dey, "Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns." vol. 2003, 2003.

[13] BuiltWith, "Google API Usage Statistics." [Online]. Available: https://trends.builtwith.com/javascript/Google-API [Accessed: 2020-11-03]

[14] 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.

[15] 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/

[16] 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

[17] StatCounter, "Mobile Operating System Market Share Worldwide." [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide [Accessed: 2020-01-15]

[18] The PHP Group, "PHP." [Online]. Available: https://www.php.net/ [Accessed: 2020-11-03]

[19] PgAdmin, "pgAdmin Frontpage." [Online]. Available: https://www.pgadmin.org/ [Accessed: 03-11-2020]

Appendix A

Additional Files and Schematics

Add any information here that you would like to have in your project but is not necessary in the main text. Remember to refer to it in the main text. Separate your appendices based on what they are for example. Equation derivations in Appendix A and code in Appendix B etc.

Appendix B

Addenda

B.1 Ethics Forms