LITTLE LOCATOR: AN INNOVATIVE SMARTPHONE PARKING RELOCATION APPLICATION

Joseph Lewis

Bachelor of Engineering
Software Engineering

MACQUARIE
University
SYDNEY · AUSTRALIA

Faculty of Science and Engineering
Macquarie University

November 6, 2017

Supervisor: Dr Robert Abbas

ACKNOWLEDGEMENTS

I wish to acknowledge my project supervisor Dr Robert Abbas for the initial conception of this project and mentorship throughout the duration of the semesters I undertook my research. I would also express my gratitude for his help in my development as I become a professional engineer.

STATEMENT OF CANDIDATE

I, Joseph Lewis, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Software Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment at any academic institution.

Student's Name: Joseph Lewis

Student's Signature: J Lewis

Date: 6/11/2017

ABSTRACT

Smartphone applications that save the locations of where vehicles are parked aid users in relocating their vehicles so that individuals are not required to rely solely on their memory and backtracking skills. These types applications typically utilise GPS chips in devices and make use of mapping APIs to place markers on the map for straightforward navigation back to the vehicle's position. The GUI must be developed to integrate seamlessly with the functionality of the application, particularly focussing on facilitating the fulfilment of the requirements elucidated for the human-computer interface. This research aims to analyse the current state of smartphone parking applications, determine the design and functionality for optimal usability, as well as complete testing and documentation to ensure professional standards are being met, progress can be tracked and users can understand the system. Value has been added to the parking relocation application, named Little Locator, by adding innovative features which competitors have not conceived and focusing on user experience. These features include a local background service that pinpoints when a user has parked their vehicle and navigation for the visually impaired that utilises computer vision.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

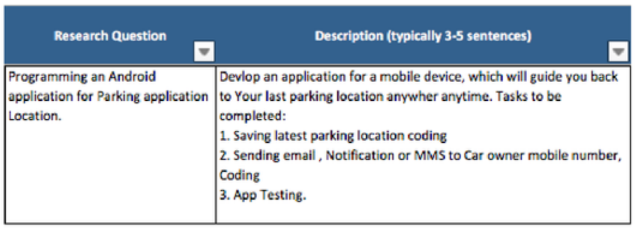
The primary objective of a smartphone parking relocation application is to maximise the efficiency of returning to a user's parked vehicle. Individuals have a tendency to forget where they have parked when they are preoccupied with other matters or are in an unfamiliar area. This project goes beyond current solutions and utilises new techniques that analyse the movements of users to pinpoint where their vehicles are located if they forget to manually set their parking location in the app, as well as adding navigation functionality for visually impaired users. The result of this research project will particularly benefit the elderly and visually impaired as the interface has been designed specifically for simplicity. This final report defines the research problem, background information and existing research in the area, discussions of the design and implementation of the solution, plausible outcomes and conclusions drawn over the period the research and development was conducted.

Additionally, this document contains discussions the current state of the project, and future work related to the product of that research.

1.1 The Problem

Recalling where one has parked their vehicle after a period of time, can be a challenge for most people at various times in their lives. Little research has been completed to gain statistics on the number of people and frequency of which this phenomenon has occurred, apart from a limited number of unverifiable surveys that have been conducted. Nevertheless, these surveys would not be considered reliable sources. An individual forgetting where they have parked can be a significant problem, as the temporal cost of locating vehicles should not be a requirement for individuals that have forgotten the location of their vehicle after a period of being occupied with other tasks. Individuals in a rush may also be late to their next obligation, where the forgetfulness will be amplified if they become flustered. One solution to this problem is utilising a smartphone that uses location services to place a marker where the user has parked, allowing the individual to navigate back to their vehicle. Multiple software solutions are already available on the Google Play Store, Apple App Store and on the internet findable using a search query. However, analysing the unique download counts of such apps indicate that utilisation of these apps remains relatively low; none of these applications appear on 'top downloads' charts [1], [2] and the highest displayed applications do not even reach 10,000 downloads. This indicates that the current solutions available are not well-known or are of suboptimal effectiveness.

Prior to the state of the art implementations discussed, there have been few-to-no solutions based primarily on relocating a parked vehicle. This is due to smartphone developments making it possible to develop convenient and versatile applications for uses using GPS location services and other available APIs.



Research Question	Description (typically 3-5 sentences)
Programming an Android application for Parking application Location.	Devlop an application for a mobile device, which will guide you back to Your last parking location anywher anytime. Tasks to be completed: 1. Saving latest parking location coding 2. Sending email , Notification or MMS to Car owner mobile number, Coding 3. App Testing.

Figure 1.1 – The research question

The question this research aims to answer is how to deliver a smartphone software solution that addresses the research question and provide an innovative approach in functionality and UI design. The purpose of this project is to create a solution that mitigates the issue of drivers wasting time searching for their vehicles when they are attempting to return to them. This phenomenon typically occurs after the individual has not given thought to where they parked, or have been preoccupied with other tasks. The research that has been conducted has been a crucial part of the project as it gives insight into what solutions already exist, promotes innovation and leads to a product that stands out by being highly user-friendly and incorporating new, helpful features. Even after the primary stage of research current advancements in software features and technology were assessed throughout the year to ensure the latest methods were used and new ideas could be provoked. However, once the initial research had been completed development of the application used the majority of the allocated time.

1.2 Objectives

The primary objective of this project was to research current implementations of parking applications analyse possible innovations to deliver a usable Android application that facilitates the storage of users parking locations, and allows recollection based on the retrieval of that position. A notification was to be configured to remind users of when they must return to their vehicle, and comprehensive testing has been completed to ensure that this primary functionality is correct. It was expected that this objective was to be resolved early into the project and optimising the application, improving the user interface, and adding miscellaneous functionality to improve the user experience was to be the main focus of the project.

The project was divided into activities with associated short-term goals, long-term goals and additional non-goals. The short-term goal of the project was to satisfy the primary objective, which has been completed. This short-term period constituted of the time spanning from the start of session 1, 2017, to 6 November, 2017.

The non-goals include extending the functionality of the application by implementing long-term goals elucidated into the project. A number of these non-goals will be implemented during the short-term period, where time permits. Routing and navigation were non-goals conceived during the development of the project plan delivered in session 1. Routing functionality has been successfully implemented, while focus on other complex non-goals resulted in navigation not yet being implemented.

A complex and innovative non-goal was to incorporate machine learning (statistical and predictive analytics) to recognise the general areas that a user typically parks, with algorithms to detect travel changes in an attempt to find their vehicle if the user has forgotten to manually set their parking location. This functionality was non-crucial to the project, but added significant value to the final project.

A collection of additional features to assist the visually impaired in using the application was another non-goal in this project. Implementations based on current plans utilise functionalities including voice recognition and text-to-speech technology. This non-goal can be extended to allow visually impaired users to navigate through routes in a way that other navigation applications have not provided. This allows the provision of high-accuracy verbal directions and allow for detection of obstacles using Terrain, Elevation and StreetView methods in the Google Maps API, with the possible aid of Google Vision API for image analysis with StreetView. The completion of this navigation functionality was not expected to be fully-implemented short-term, although it was feasible to have a prototype or alpha implementation ready for presentation in the designated period.

A formal long-term non-goal is to develop the application for use with iOS, Android OS' main competitor in the mobile phone market. This is still a long-term goal and cannot be feasibility implemented by the deadline.

Chapter 2

Literature Review

2.1 Introduction

The purpose of this study is to research the functionality and design of similar solutions to the research problem identified in this report. This will result in additional background knowledge being gained, useful for designing and implementing the solution. Knowledge of what has already been done and what elements of design have been effective allow for creating a product, where conscious design can take place to ensure the application is not similar to what is already on the market, as well as to evoke thought on what innovative features could be useful. The background for this research topic can be reviewed in chapter 1.1.

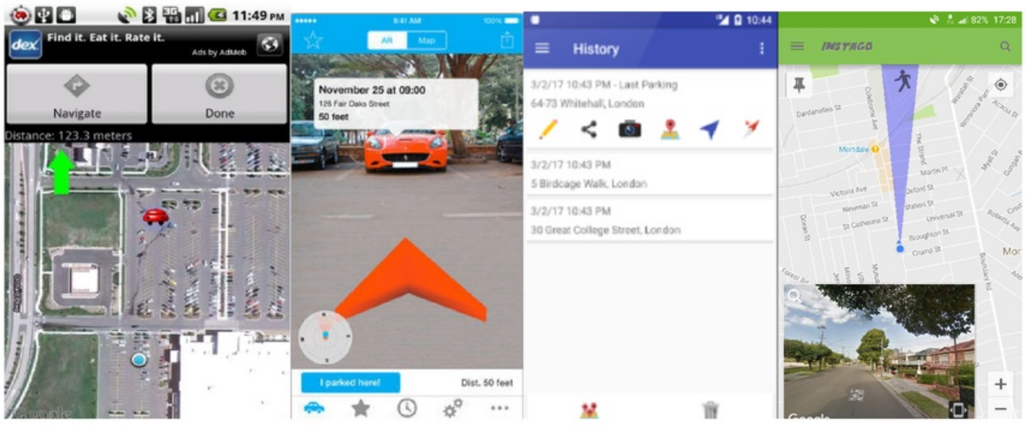


Figure 2.1 – Screens of a parking location shown on a map [6] (left), a compass-like feature leading to a parked car [4] (centre-left), showing supplementary functionality to take notes and take a photo of a parking location [7] (centre-right), and a screenshot from Instago [12] (right)

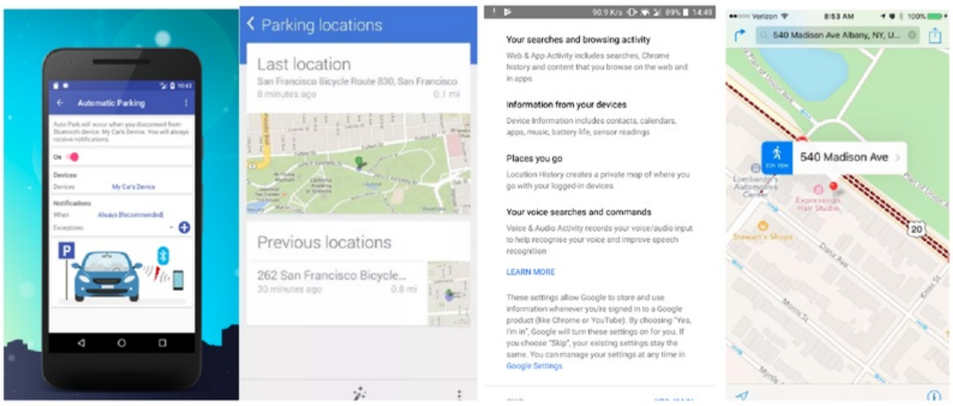
2.2 Functionality

The top hit search results related to lists of Android and iOS parking applications, shown in [3] – [5], contain several features that are predominant in a high number of third-party applications that assist a user in locating their parked vehicle. Additional applications on the Play Store and App Store share these common trends in functionality [6], [7]. Common functionalities in many of these applications include:

- A map with a marker that points to the location of the parked vehicle to allow users to navigate back to it. It should be noted that only one of the applications viewed provided walking navigation back to the vehicle. Car Compass provided this functionality, although it has a very outdated design and limited additional functionality [6]. The navigation screen for this application is displayed in the far-left image of figure 4.2. Additionally, all of these applications require an internet connection to use this feature.
- A compass-like view that utilises the GPS system to provide a pointer that shows the general direction and distance of the parked vehicle while the user is offline, as shown in the middle screen of figure 2.1.
- Supplementary features; an option to take a photo of the environment where the vehicle has been parked, the ability to write a note, and functionality to set a reminder of when the user must return to the vehicle, shown in the far-right screen in figure 2.1.

One of the only available applications that can “automatically” locate a user’s vehicle, works by saving the location of the device when the user disconnects from their vehicle’s Bluetooth system. This functionality has been implemented by ParKing Premium for Android [8]. Apart from this feature, the application is similar to the other applications studied. This method is effective if the user regularly pairs their device with their vehicle’s media system. However, many vehicle owners do not have or use Bluetooth pairing, so another solution is required. Another solution available in some luxury vehicles work using a GPS chip in the car that can be used by a smartphone application to locate the car from anywhere with a signal. This feature is accessible to even less people.

Recently, Google and Apple have been on features that save where Android and iOS users have parked, displayed in Google Maps or Apple Maps respectively [9] – [11]. Google’s solution is implemented in Google Now and uses a location tracking service that can estimate the proximity that a user could have parked, but only if they have enabled the feature through accepting numerous data collecting agreements. This functionality has several downsides. The core functionality of Google Now is not to locate vehicles – for this reason the feature is not that straightforward to use, users have no control of this feature other than turning it on or off and with all of Google's other projects, Google has not placed that much focus on this feature. Apple's solution is very primitive relative to Google's solution. Additionally, uses the functionality of Google Now requires a lot of user trust, as a lot of personal identifying user information is stored on Google servers. The feature was only introduced in 2017 and the users parking location is only stored if they used Apple Maps to navigate to a destination. That destination is set as the location of the vehicle. Both pins can be placed, as well as notes added with the solutions by Google and Apple.



Your searches and browsing activity

Web & App Activity includes searches, Chrome history and content that you browse on the web and in apps

Information from your devices

Device Information includes contacts, calendars, apps, music, battery life, sensor readings

Places you go

Location History creates a private map of where you go with your logged-in devices

Your voice searches and commands

Voice & Audio Activity records your voice/audio input to help recognise your voice and improve speech recognition

LEARN MORE

These settings allow Google to store and use information whenever you're signed in to a Google product (like Chrome or YouTube). By choosing "Yes, I'm in", Google will turn these settings on for you. If you choose "Skip", your existing settings stay the same. You can manage your settings at any time in Google Settings.

Figure 2.2 – Screens of ParKing Premium: Find my car automatic parking feature [8] (left), parking location functionality on Google Now [9] (centre-left) and Apple Maps [11] (right), and Google Assistant asking for privacy permissions when launching the application (centre-right)

Instago Street View Navigation [12] provides unique functionality that displays Google StreetView images in a window that updates according to the user's GPS position. It allows users to place a marker at a location and navigate to it on a map view.

2.3 Design Features

In the same collection of researched current solutions [3] – [7], there are an array of design features implemented in the presentation of the applications. One typical method of providing parking location functionality is to display a map view with an overlay of buttons to set where the car was parked. An alternative approach is to have a default layout that allows for setting the parking location and any other additional information, before being redirected to the map view when returning to the vehicle. Toolbars in Android, or Navigation Bars in iOS, can be used to switch between multiple layouts in a user-friendly manner.

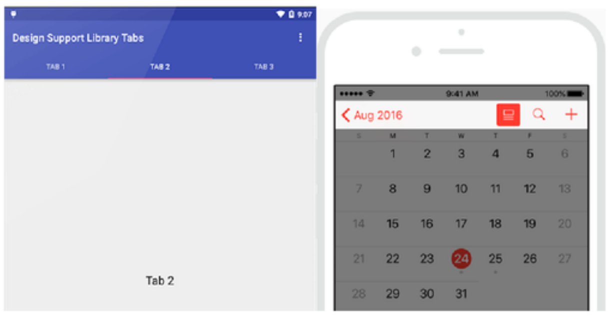


Figure 2.3 – Examples of the Android ActionBar [13] (left) and iOS Navigation Bar [14] (right)

Several of the applications used text boxes to receive input from the user regarding additional parking information, while other uses used built-in objects including TimePickers, NumberPickers and camera previews on SurfaceViews. This allowed for less reliance on buttons and text fields to complete actions in an ergonomic style.

2.4 Conclusions

During acceptance testing of these applications, it was evident that the applications that relied less on text boxes for input felt more fluid for the user. Utilising tabs at the top of the device's screen added to the ease-of-use for these applications. During the completion of the project, these findings should be considered when undertaking the design and development of the solution. Adhering to the design guidelines set by Google and Apple also provide a greater feeling of modernity and integration with the OS. For this reason, attention to design is integral and the most recent stylings and objects should be used. Alternatively, to increase backwards compatibility, choosing to style the application to the “device default” theme would be a beneficial option.

It is integral to include a screen that shows a map of the vehicle’s parking location to provide the user with optimal detail of the environment. This was supported by numerous acceptance tests of the researched applications. Additional functionality such as a compass or simplified view could also be beneficial to the usability of the application. This research shows that applications with unique features stand out from the competition. Additional features that are not gimmicks provide value to the application, giving more incentive for users to use that product. During the conceptual design stage of development, contemplation of innovative features should be a priority. Examples include the majority of the non-goals elucidated in chapter 1.2. The functionality exhibited in Instago highlights the feasibility of utilising or analysing Google StreetView image data to innovatively aid visually impaired users in navigating back to their parked vehicles. This StreetView data could also be stored in a SurfaceView if the user does not choose to take a photo of their location when they park.

Chapter 3

Survey Methodology

3.1 Overview

The literature review conducted provided useful insight on functionality and user experience. These two insights are integral to producing a well-informed solution to the questions posed by this research. The research needed to be extended to include unprecedented experimental investigation to ascertain where the public stands in regards to relocating their parked vehicles, what they would benefit from and the types of devices they primarily use.

An online survey created for this purpose was provided to willing participants during the later stages of research and through the earlier stages of development. The results from these surveys were used as a tool to design a solution that would benefit the types of individuals that would use the solution produced.

3.2 Survey Questions

The questions available in the survey were conceived to answer questions specifically relevant to designing a solution, and additional questions were added once the investigation had started as new insights emerged. The questions in the investigation included:

- Do you often remember where you park your car in new places? (Scale of 0-10)
- Do you often remember where you park in a large parking lot or shopping centre? (Scale of 0-10)
- Would you use a smartphone application to help you navigate back to your car? (Yes, Unsure, Maybe)
- How helpful would it be for the application to run in the background to automatically detect when you park? (Scale of 0-10)
- What type of phone do use primarily? (Android, iPhone, Blackberry, Other smartphone, Non-smartphone, No phone)
- Would in-app navigation functionality be useful to you? (Yes, No)
- Have you used any of the following applications to help you remember where you parked? (Google Now, Apple Maps, 3rd party parking application, None)
- What is your age?

The following questions were added once the implementation of the initial design was underway and related to what type of user experience people wanted and whether online privacy was a concern for them. Online privacy related to the data collected by the location service that could determine to a high degree when a user had parked their vehicle. The data was stored locally on the device, although input was appreciated to whether users would mind if anonymous data was sent to a central server to improve the accuracy of the feature that could pinpoint when a user had parked, named WHERE.

- Are there any additional features that you would find useful? (optional)
- How important is online privacy to you? (Scale of 0-10)
- Would you like more of a focus on functionality or simplicity? (Scale of 0-10)

Once a considerable amount of data was collected, the results of the survey were analysed to gain insight into the public trends related to parking and what the focuses on design and implementation should be. The complete survey can be seen in appendix A.

3.2 Recruitment Methodology

Participants were known to the author, and efforts were made to select candidates of varying age groups in an attempt to eliminate biased results. 25 candidates were asked to complete the survey via a static link, and results would automatically be analysed in a results page. The medium used for this survey was https://freeonlinesurveys.com. Out of these 25 candidates, there were 23 recorded participants.

Chapter 4

Survey Results

The format of the survey allowed for quantitative results to be received, with qualitative aspects formatted as sliding scales to gain standardised results. This would be comparable to a scale used by a doctor that wishes to know the amount of pain their patient is in. These results both supported the validity of the value associated with the research problem and promoted focus on areas relating to participant feedback. Conclusions will not be drawn from the results of a single question, but by analysing the results of the survey as a whole.

4.1 Relocating a Vehicle in a New Area

A sliding scale was used for answering this question. Out of the 23 responses on average 38.4% of the time people had trouble relocating their parked vehicles in new area. This is shown in figure 4.1.

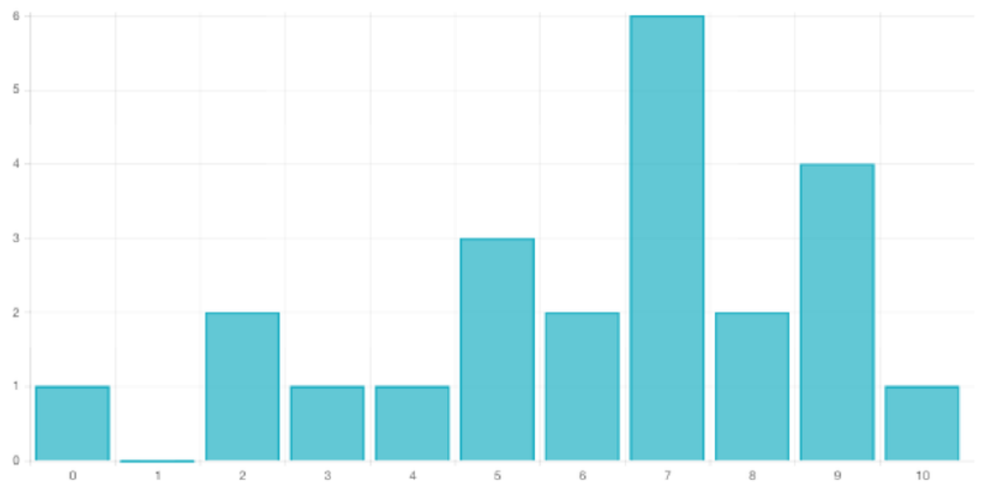


Figure 4.1 – Results from question 1 of the survey, the y-axis shows the number of answers

The results show that out of the sample collection at 1 out of 23 have never forgotten where they have parked in an unfamiliar area, and the majority of people have forgotten where they have parked 5 out of 13 times. While there are some outlying results, taking the average of the data is an effective method of obtaining a general trend.

4.2 Relocating a Vehicle in a Large Parking Lot

Question 2 of the survey was structured identically to question 1. Out of the 23 responses on average 43.5% of the time people had trouble relocating their parked vehicles in new area. This is shown in figure 4.2.

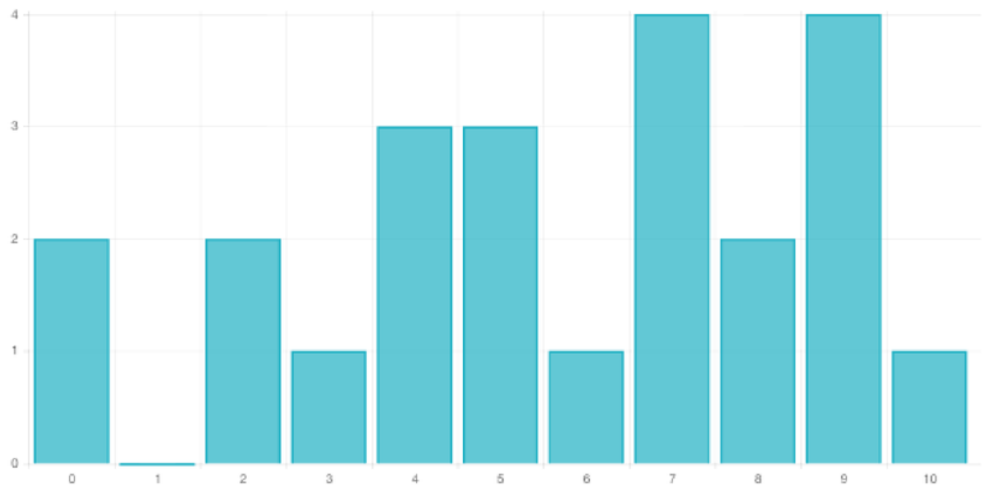


Figure 4.2 – Results from question 2 of the survey, the y-axis shows the number of answers

It was unanticipated that more people out of the participants reported that they had more trouble remembering where they parked in a shopping centre or parking lot compared to a new area.

4.3 Demand for a Smartphone Parking Application

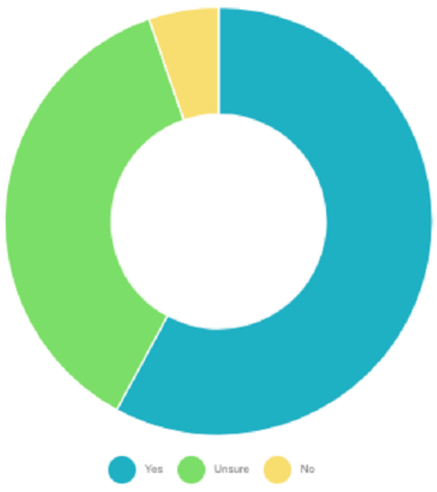


Figure 4.3 – Results from question 3 of the survey

19 responses were given for question 3, asking if participants would use a smartphone application to help them navigate back to their vehicle. 11 (58%) of the respondents said they would, 1 (5%) said they would not and 7 (37%) said they were unsure if they would use such an application. It is evident that the majority of people asked would either use or be open to using a parking application to help them relocate their vehicle.

4.4 Value of Automatic Parking Detection

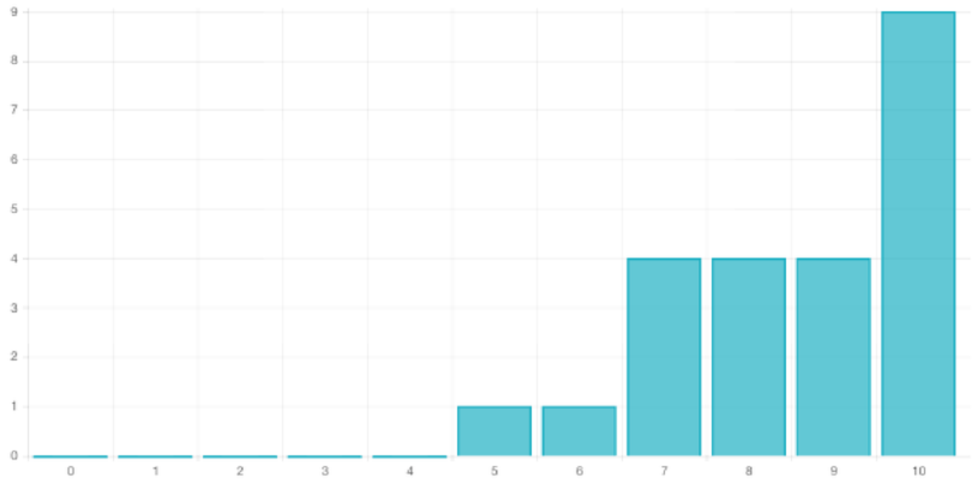


Figure 4.4 – Results from question 4 of the survey, the y-axis shows the number of answers

85.7% of survey participants affirmed that it would be helpful if the parking application could run in the background to automatically detect when they park. There were no results where a respondent was under 50/50 sure whether such a feature would be useful.

4.5 Types of Phones Primarily Used

13 (53%) of the respondents to this question primarily used an Android smartphone as their daily driver. iPhone users followed with 9 (39%) of participants owning one. Only 1 (4%)

participant did not own a smartphone, and none of the respondents primarily used a Windows or other smartphone, nor no phone at all.

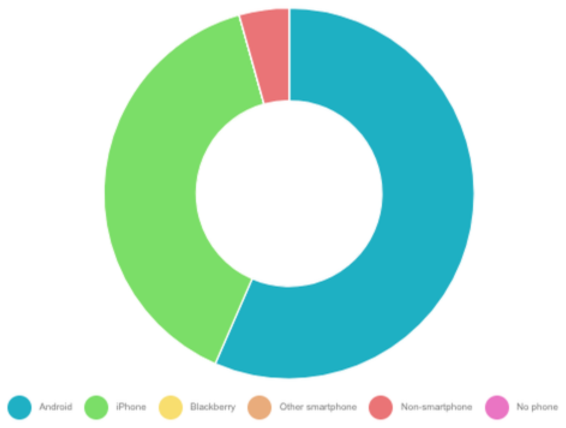


Figure 4.5 – Results from question 5 of the survey shown on a pie chart

From this data, a conclusion can be quickly drawn that it is only worth-while to develop such an application for Android OS or iOS when applying a typical business model. This is clearly illustrated in figure 4.3.

4.6 Demand for In-app Navigation

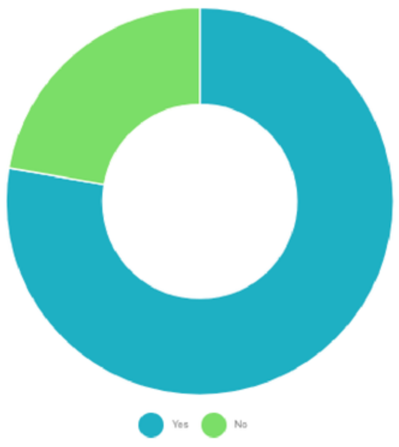


Figure 4.6 – Results from question 6 of the survey shown on a pie chart

14 (78%) of respondents to question 6 submitted that they would benefit from in-app navigation in a parking application, while only 4 (22%) that such a feature would not add value to the application.

4.7 Usage of Similar Alternatives

Out of the 18 respondents to question 7, 4 (22%) people said they had used the feature in Apple Maps to place a marker where they had parked and 1 (6%) of the group said they had used Google Now to help them relocate their vehicle. No participant had used a third-party application and 13 (72%) of the respondents had never used an application on a smartphone for vehicle relocation.

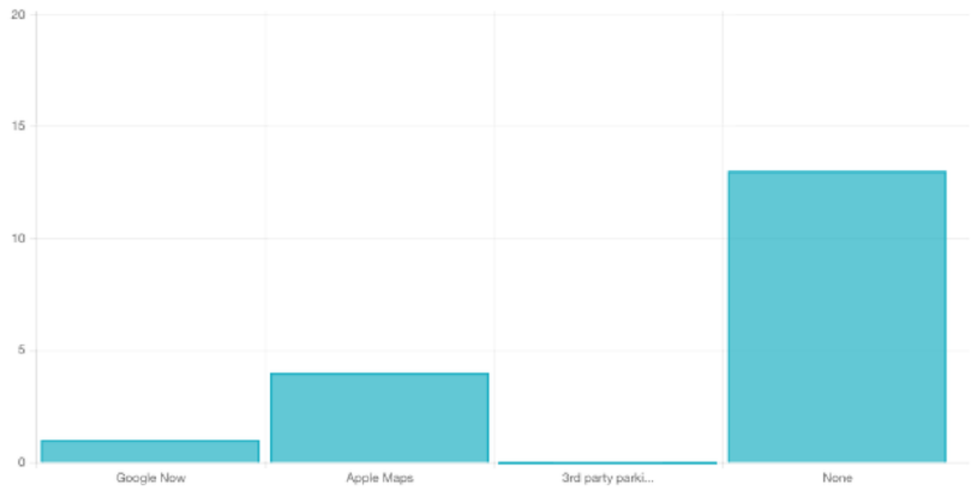


Figure 4.7 – Results from question 7 of the survey, the y-axis shows the number of answers

4.8 Age of Participants

From the 23 survey participants, 2 were 18, 1 was 19, 4 were 21, 5 were 22, 2 were 23, 1 was 25, 1 was 27, 1 was 28, 1 was 35, 1 was 37, 1 was 38, 1 was 50, 1 was 51 and 1 was 59. For simplification, the ages have been placed into categories shown in table 4.1.

Table 4.1 – Survey participants of driving age, grouped by decade

Age (Yr)	16-19	20-29	30-39	40-49	50-59	Average (Yr)
Total	3	14	3	0	3	28.04

The main purpose of this question was to identify what generations the results represent.

4.9 Additional Comments from Participants

There were 3 anonymous responses for question 9 that asked participants to write a comment on a feature that they would find useful. This was stated as a non-compulsory option for respondents. The comments are as follows:

- “I didn't know there were parking apps available.”
- “Help you catch public transport”
- “help you find parking spots in the area”

4.10 Importance of Online Privacy

On another sliding scale, 6 (46%) of respondents affirmed that their online privacy was of the upmost importance, and 6 (46%) highly prioritised it. Only 1 (8%) of participant held online privacy at a low regard. There were less answers for this response as the question was added once the project had already commenced. The results lead to the fact that participants believe that online privacy is of 86.9% importance.

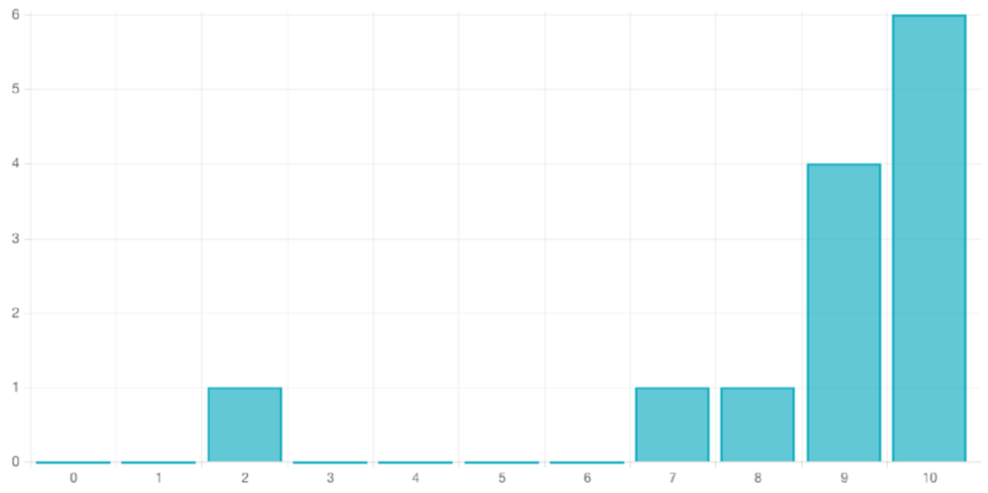


Figure 4.8 – Results from question 10 of the survey, the y-axis shows the number of answers

While most people say online privacy is important to them, from their characters the author knows that they do not make an effort to ensure their online privacy to a standard that matches their responses. This is a serious issue, as technological users should be aware of the data that is being collected by service providers and other online entities.

The responses from this question infer that the collection of the data specified in the EULAs of Google and Apple stores from the users of their products would be unwanted and considered intrusive, whether the respondents are aware or not.

4.11 Functionality Versus Simplicity

The final question on the survey related to whether participants wanted simplicity or functionality in such an application where they could respond on another quantitative sliding scale. The results for this question was by far the most even, with the average response saying that users want a 52.3% focus on simplicity, with the rest of the focus on functionality.

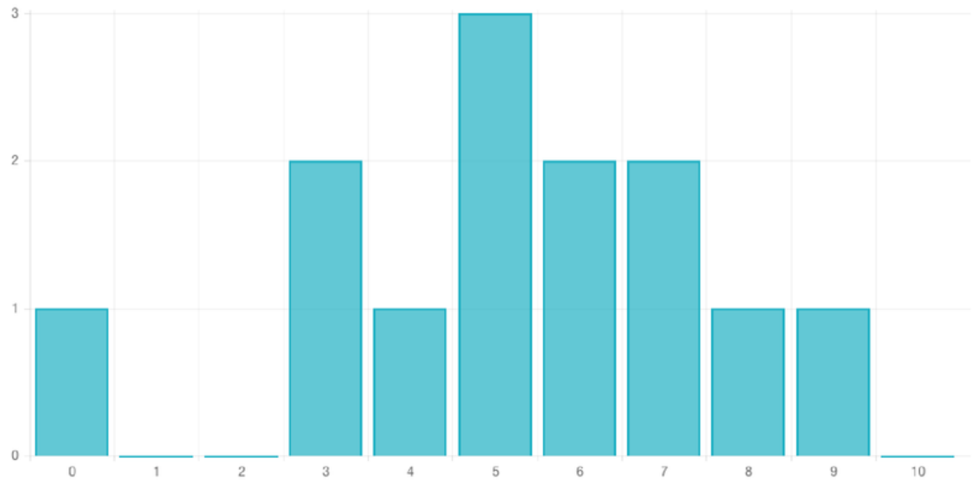


Figure 4.9 – Results from question 10 of the survey, the y-axis shows the number of answers. Results further to the left are functionality-focused, while the right indicates a focus on simplicity.

It can be inferred that users want a balance between simplicity and functionality, where the product is not too feature-heavy or “advanced”, but is not too simple either.

4.12 Conclusions

Each question asked was related to a particular area of the research problem. The first two questions provided data to support the assumption that there was a need for such a solution to the perceived problem. From the analysis of the results it can be concluded that from the data it can be assumed that there is indeed a problem that needs a solution.

It can be drawn from question 3 to 7 that the majority of users would have an Android or iPhone smartphone, that the majority of people have not utilised a parking application for relocating their vehicles and that automatic parking analysis and navigation are desired features for most individuals.

The additional questions appended to the survey once development had commenced affirmed the assumptions of the author that privacy has significance to individuals, and that the general audience of the product would benefit from a balance between functionality and simplicity. This was the aim when designing the Android application for usability.

Upon reflecting on the comments added by survey participants, neither of the two suggestions were practical to implement in the project. Helping uses find car parking spots would be a major update that would require more time and resources than allocated for this project, whilst transportation navigation is out of the scope of the application. In addition, adding too many varying features would diminish the simplicity that users wish to have in an application.

The conclusions drawn from this survey were effective in validating the appositeness of the direction of the research progress. An important enhancement of the project would be to develop the application for compatibility to iOS because as of Q1 2017, 14.7% of the smartphone OS market share belongs to iOS [15].

Chapter 5

Design and Implementation

5.1 An Agile Approach

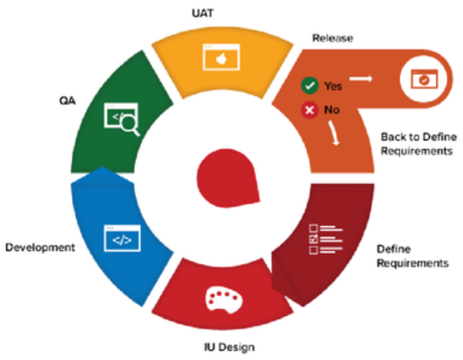


Figure 3.1 – An example of an agile approach iteration [16]

An agile approach was taken when developing the solution. This type of approach works by defining requirements to create a conceptual design, then designing and developing software based on those requirements. The completion of QA testing, delivering the features and ascertaining feedback for each iteration are all part of the process [16]. These iteration loops repeat for each feature or story that is required to be integrated into the project. This allows for adaptive development based on feedback, as well as speed due to the short fixed-time periods set for each iteration. This approach creates visibility of complete working features much earlier in the project life cycle compared to a plan-drive approach. This provides insight for gauging progress and quality, allowing for adaption during the project development cycle, especially when working as an individual developer.

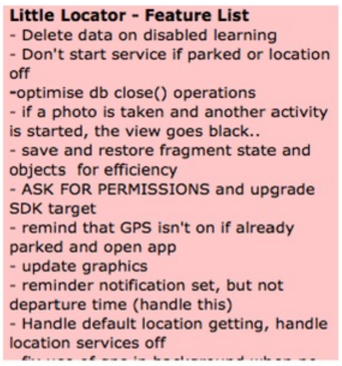
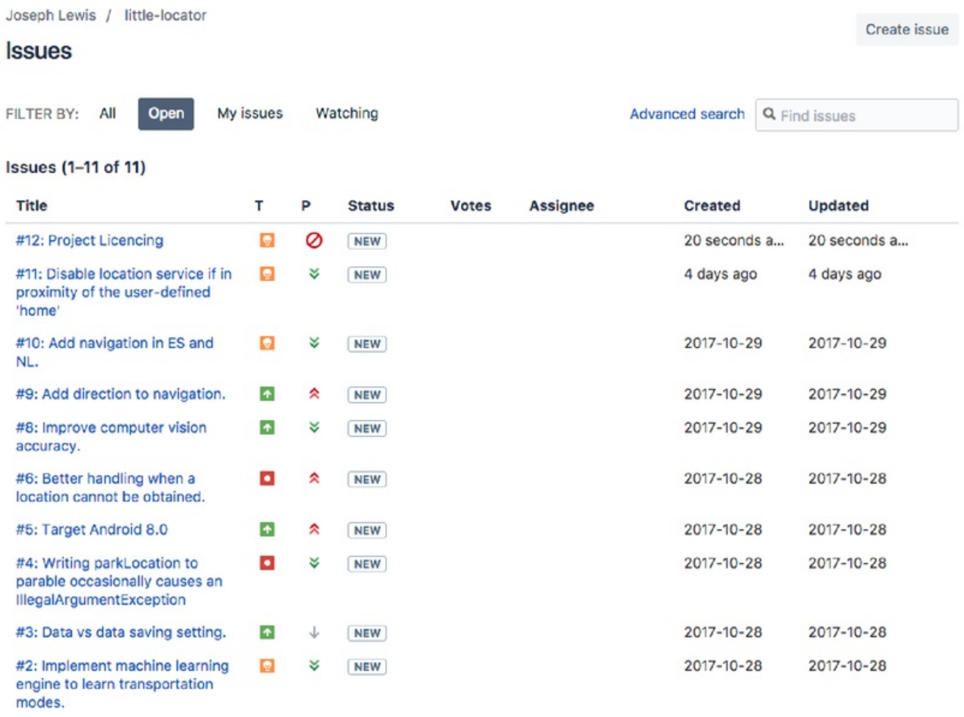

Little Locator - Feature List
- Delete data on disabled learning
- Don't start service if parked or location off
-optimise db close() operations
- if a photo is taken and another activity is started, the view goes black..
- save and restore fragment state and objects for efficiency
- ASK FOR PERMISSIONS and upgrade SDK target
- remind that GPS isn't on if already parked and open app
- update graphics
- reminder notification set, but not departure time (handle this)
- Handle default location getting, handle location services off

Figure 5.2 – Rough feature list for the project


Joseph Lewis / little-locator

Issues

Create issue

FILTER BY: All **Open** My issues Watching   Advanced search Find issues

Issues (1–11 of 11)

Title	T	P	Status	Votes	Assignee	Created	Updated
#12: Project Licencing	🟧	⊘	NEW			20 seconds a...	20 seconds a...
#11: Disable location service if in proximity of the user-defined 'home'	🟧	⏬	NEW			4 days ago	4 days ago
#10: Add navigation in ES and NL.	🟧	⏬	NEW			2017-10-29	2017-10-29
#9: Add direction to navigation.	🟩	⏫	NEW			2017-10-29	2017-10-29
#8: Improve computer vision accuracy.	🟩	⏬	NEW			2017-10-29	2017-10-29
#6: Better handling when a location cannot be obtained.	🟥	⏫	NEW			2017-10-28	2017-10-28
#5: Target Android 8.0	🟩	⏫	NEW			2017-10-28	2017-10-28
#4: Writing parkLocation to parable occasionally causes an IllegalArgumentException	🟥	⏬	NEW			2017-10-28	2017-10-28
#3: Data vs data saving setting.	🟩	↓	NEW			2017-10-28	2017-10-28
#2: Implement machine learning engine to learn transportation modes.	🟧	⏬	NEW			2017-10-28	2017-10-28

Figure 5.3 – Extract of issues logged in the Little Locator Bitbucket repository

As numerous goal and non-goal objectives were defined during the earlier stages of the project, the methodology has aligned with the agile branch of FDD. The FDD process began by developing an initial high-level model of the application. This model contained a schematic for the interface and actions available when a user parked and then returned to their car. The next stage was to create a feature list, where a single feature would be implemented in one iteration. The project contains more structure than this basic model.

However, as there was a clear vision, implementing the objectives using FDD has been an effective approach.

As the project progressed, features were continually elucidated and written in rough notes on the project workstation as seen in figure 5.2. Professional practice increased throughout the duration of the year, and eventually features and bugs were listed as issues on Bitbucket. This was necessary due to increased complexity of the code as the application evolved and worked well with version tracking. An extract of the recent issues is shown in figure 5.3. This method was effective as compartmentalised improvements could be made, ensuring a working prototype was available at all times, and recoveries were simple due to features and improvements that were being implemented being recorded to a changelog with timestamps.

Story points were assigned to each feature, improvement or bug fix depending on the perceived time that would be required to resolve the issue. This meant that between each revision, typically 2-5 issues were resolved. The complete revision history can be viewed in appendix B. The resultant Android application has been titled “Little Locator”, under the package name com.littleinnovations.littlelocator.

5.2 Activity Statements

The project was broken down into a list of activity statements for macro-level tasks. As an agile approach has been taken, steps 2-4 had the longest timeframe as they have been repeated for each of the features that have been implemented in the application. As such a long period has been spent on development, research, testing and writing the report have been completed concurrently. The following activity statements provide an overview of the project as a whole:

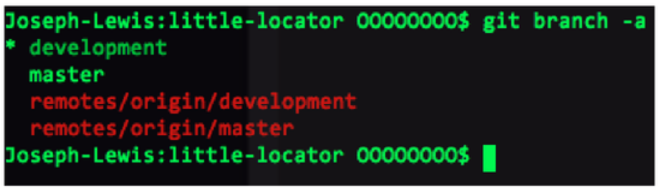
1. **Research** – A study of similar solutions available and the technology in existence to help create an innovative solution to the research problem. This activity included a literature review on the topic, which can be viewed in can be viewed in chapter 2. The literature review has been expanded throughout the duration of the research project. The research led to a development of knowledge and skills as an Android platform developer to create a better and more efficient application. The results of all the

research are viewable throughout this document, as the research continued past the stages literature review and survey were conducted.

2. **Conceptual Design** – The quantitative and qualitative requirements for the software solution are defined to help create a framework for the application. The requirements specified will be aided by the research conducted in the previous activity that has defined the problem. A feasibility study for components, system trade study for implementation methods and system planning have been carried out prior to and during this research component.
3. **Design and Development** – Detail design will be carried out in this activity, along with the development of a fully functional application that satisfies the requirement goals. Testing will occur in this activity that will be used to fix bugs and allow for modifications for corrective action. Test cases will be created before the development of the application to ensure the testing is more thorough and contains minimal bias. System analysis and evaluation will occur to allow for improvements related to the usability and functionality of the application. An agile methodology has been utilised; hence there will be multiple working prototypes delivered after each sprint.
4. **QA Testing –** Intensive testing will occur in this stage to ensure the product is usable, meets all requirements, responds correctly to inputs and performs functions in acceptable timeframes. Where applicable, corrective actions will be taken during this stage in the project lifecycle. Feedback will also be obtained and actioned in this stage, amending requirements and taking corrective action in line with the agile methodology.
5. **Finalising the Research Report –** The research report was in progress throughout the entire project. The thesis had been a week in progress until the application had been completed to a desired standard.
6. **Presentation** – A presentation and poster have been created to accompany a demonstration of the application developed, to highlight the work completed in this project. At this time, an abstract, logbook and administrative documentation are also completed.
7. **Maintenance** – Optionally, after the initial solution has been delivered, the project will have entered the maintenance phase, updating components when necessary and ensure that the application is compatible with future APIs and OS versions. Additional functionality can continually be implemented along with bug fixes.

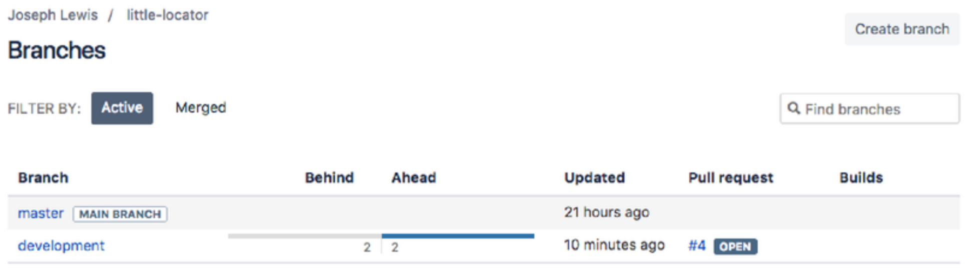
5.3 Development Environment

The Android Studio IDE was used for the implementation of the solution, along with Bitbucket for version control starting at version 0.92. The application is written in Java. Prior to Bitbucket version control, a local repository was set up for version control as for individual work that was efficient. Android Studio and Git are free, currently maintained and available on all three of the main OS distributions, Linux, MacOS and Windows, so these options were suitable choices.



```
Joseph-Lewis:little-locator 00000000$ git branch -a
* development
  master
  remotes/origin/development
  remotes/origin/master
Joseph-Lewis:little-locator 00000000$ 
```

Figure 5.4 – Working in development branch of the project



Joseph Lewis / little-locator

Branches

Branch	Behind	Ahead	Updated	Pull request	Builds
master MAIN BRANCH			21 hours ago		
development	2	2	10 minutes ago	#4 OPEN	

Figure 5.5 – Merge development in master branch once ready

All development was completed in the development branch shown in figure 5.4, and once testing had been completed a merge to the master branch would be made. This was the most efficient way of ensuring that while development was ongoing there would always be a stable build. If access to Atlassian's Stash was available it would be more feasible to create a branch for each story due to the issue tracking functionality, although this would be unnecessary for use by a single software engineer.

5.4 Costs

Due to the utilisation of free development tools and APIs for mapping capabilities, there were no associated costs related to the project itself. The author attended ANZSCON 2017 for project related reasons that will be discussed in appendix C.1, which cost $220 and was sponsored by the Faculty of Science and Engineering Macquarie University.

5.5 Application Features

Thorough research has been conducted on each of the objectives in the scope of this project to provide a well-designed solution to the research problem. The list is an overview of the majority of the functionality and novelties of the application, expanding past the objectives discussed in chapter 1.2.

- Saving users parking locations, and pinning the coordinates on a map.
- Configurable notifications for when the user must depart.
- A fluid, interactive interface; a photo can be taken of the parking environment, reminders can be set for when a user must return to their vehicle and notes can be added to the parking location.
- A reverse geocoding feature, allowing a readable address to be associated with the parking spot.
- An offline GPS compass that shows the distance and direction back to the vehicle. The user can set the background to a camera preview for additional visibility when navigating.
- The ability to enter a custom location for setting a route. Walking directions have been implemented with algorithms to work out the distance and estimated time the journey will take. A speedometer that uses the GPS chip in the device gauges the speed of the user.

- WHERE: a feature where algorithms can estimate where the user's vehicle is parked if they have forgotten to manually set their location. This is a large feature that has been refined for the majority of the project to increase efficiency and accuracy. Google has been the only other company to publically attempt this. The major advantages of this implementation over Google Now are that,
    a) all user data is processed locally and user data is not sold to third-parties for targeted marketing, and
    b) This is a core feature of the application unlike with Google Now, so it will be more feature-driven and usable.
- Walking navigation that incorporates computer vision through Google Vision API to provide warnings for obstacles on the route such as poles and cars. This feature is a work in progress, and accessing the camera on the user's device is the next stage in the development of the feature.
- Miscellaneous features including menu options, a preferences page, an about page and privacy policy.

In addition to the frontend features visible to users, many background functionalities and optimisations have been made. This includes the use of asynchronous tasks and threading to keep the main thread fluid while more intensive operations are run, such as the location service that keeps track of the movements of users. The utilisation of alarms and receivers ensure the background monitoring service is battery-friendly and heavy optimisation of code to ensure efficient programming practices. Efforts have also been made to compartmentalise code, reduce the chance of errors and keep the substantial code package manageable.

See appendix B to view the revision history for Little Locator. Revisions are dated to provide a useful representation of how the project has progressed during its duration.

5.6 Application Development

Until the direction of the project has been discussed further with Dr Robert Abbas and the author of the research has made a decision the full source code will be made available at request. This section will contain information pertaining to the main aspects of the application development and insight into the features which provide a solution to the research problem and those that succeed the scope of the problem to further benefit users of the application.

5.6.1 Parking a Vehicle and Departing

The most simplistic feature of the solution is the parking location retrieval function, comprised of two fragments: one for entering information and the other for displaying a summary of the parking data. When a user opens the application when they have parked their vehicle they are welcomed by a screen which allows them take a photo of the environment in which they parked in, enter a time for a departure reminder and write a note such as the level in which they parked on or a nearby landmark. The screen also contains a button causing the activation of WHERE, which will be discussed later in the chapter. The interfaces for this functionality is shown on the left and centre screens of figure 5.6.

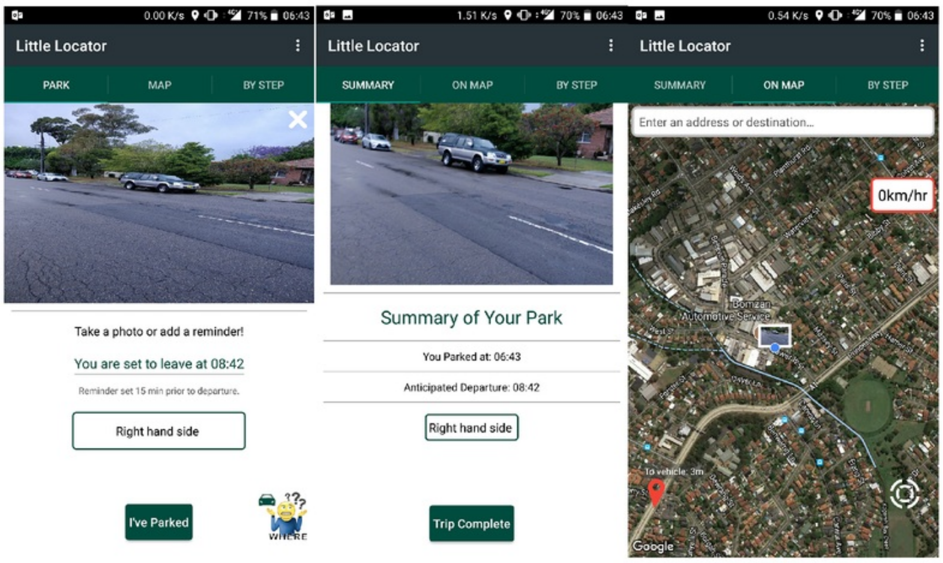


Figure 5.6 – Screenshots of the basic features that save the location of a user's vehicle

The set location of the vehicle can be viewed on the map with the photo taken set as the marker image. If no photo was taken the marker reverts to the default image bundled in the application resources. The user can navigate back to their vehicle using either the map or compass.

The parking functionality is activated in a control flow when the "I've Parked". The first check is that the device is connected to the internet, the second that the device the GPS service is enabled on the device. The parking function will work without an internet connection, but the flow will be broken and a warning message will be displayed that provides a shortcut to the device settings if GPS is disabled to prevent exception errors being thrown. After the initial checks, persistent data will be stored in SharedPreferences related to the welcome screen and device location. The data is then parsed to the summary and map fragments for reassembly. Static values are set to persist if they are to be globally accessed. A model has been implemented for creating a Parcelable parking object, that stores all the necessary information for the parking instance.

```
public class Park implements Parcelable {

    private int parkTimeHours;
    private int parkTimeMins;
    private Integer leaveTimeHours;
    private Integer leaveTimeMins;
    private String notes;
    private Location parkLocation;
    ...
```

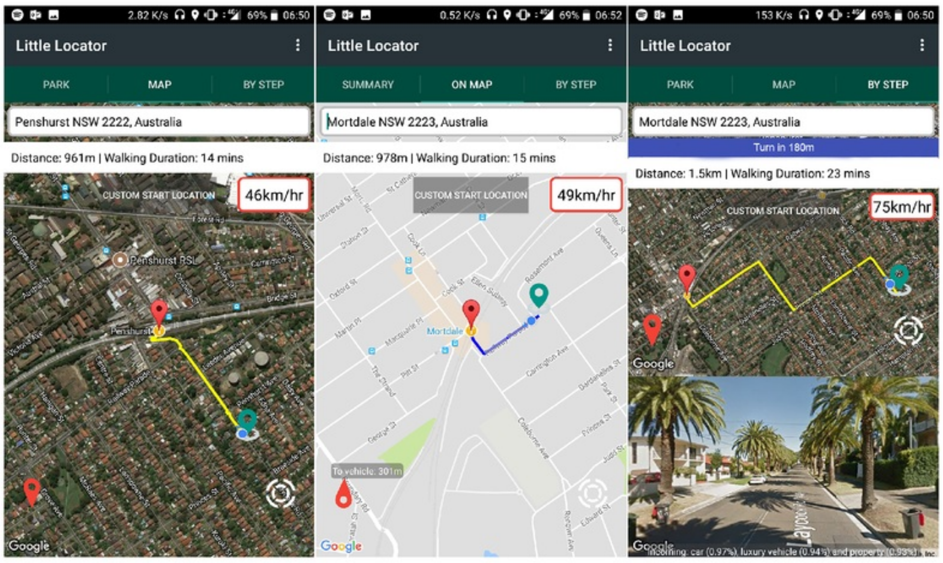
Multiple optimisations have been made to ensure a fluid user experience, including in the bitmap encoding of photos taken using multithreading and a bespoke scaling algorithm. While rare, if the image taken has not been fully processed by the time the user clicks “I’ve Parked” a toast will inform them to retry in a few seconds. An extract of the code demonstrates this functionality is shown below.

```
if (photoTaken) {
    if (cachedBitmap == null || cachedBitmapThumb == null) {
        infoText.setText("Please try again in a few seconds.");
        Toast.makeText(getActivity(),
        getResources().getString(R.string.toast_wait_photo_processing),
            Toast.LENGTH_SHORT).show();
        return;
    }
    editor.putBoolean("photoTaken", true);
    ...
```

Once the task that has processed the bitmap photos has completed, the bitmap data will be assigned to cachedBitmap and cachedBitmap thumb, hence the static variables will no longer have null values.

5.6.2 Navigation Modes

There are two layout that facilitate navigation as shown in figure 5.7. Both features are functional whether the user has a registered parked vehicle or not. The classes that provide navigation have the most code and dependencies in the project due to the complexity of implementing mapping and navigation functionality. Google Maps V2 API is integrated to display the map views, and a Google geocoding API is used to convert addresses to mapping coordinates as well as latitude and longitude pairs to human readable addresses. The other features such as the speedometer, distance to the destination and walking duration are all calculated in the application using algorithms based on mathematical principles.



Little Locator
PARK
MAP
BY STEP
Penshurst NSW 2222, Australia
Distance: 961m | Walking Duration: 14 mins
CUSTOM START LOCATION
46km/hr
Little Locator
SUMMARY
ON MAP
BY STEP
Mortdale NSW 2223, Australia
Distance: 978m | Walking Duration: 15 mins
CUSTOM START LOCATION
49km/hr
To vehicle: 301m
Little Locator
PARK
MAP
BY STEP
Mortdale NSW 2223, Australia
Turn in 180m
Distance: 1.5km | Walking Duration: 23 mins
CUSTOM START LOCATION
75km/hr

Figure 5.7 – Screenshots of the map layout (left, centre) and by step layout (right)

The speed of the device is calculated using the fact that speed is equal to displacement over time. A thread is run in the background to execute every X milliseconds while obtaining the geographical coordinates at the start and end of the execution. The speed is then calculated as:

$$speed=\frac{distanceBetween\left(location_2-location_1\right)}{x\times 3.6} \qquad (1)$$

Similarly, the distance between locations and the time it takes for an average human to walk 1km is used for calculated route lengths and walking time estimates. There is a large amount of mapping functionality, such entering custom locations for general navigation or with the use of the location of a vehicle. As the application uses Google's free commercial geocoding API, address recognition is very accurate.

By step extends the functionality of the basic map layout by adding a Google StreetView view to the screen along with TTS navigation options that can be enabled in the settings. As the user travels the StreetView images update, while the integrated computer vision API is called to analyse the image and send back objects that are parsed to give warnings to users of obstacles that may be in in their path. Additionally, the user's position relative to the route is analysed and real-time direction updates are placed in a navigation bar and spoken to the user as they travel. API calls and navigation updates are time costly, so threading and asynchronous tasks are utilised throughout these mapping classes. This can be seen in the right screen in figure 5.7, while an extract of the code that executes some of these functions is shown below.

```
mMapView.getMapAsync(new OnMapReadyCallback() { ...

latLng = new getLocationFromAddress().execute(location).get();
```

Two limitations of these features are that the direction that the user should turn in has not been implemented yet and that the obstacle analysis is not optimally tuned. Both of these improvements are easily resolvable, although due to time restraints they were not at a high enough priority at the time to be completed by the initial project completion date. These issues will be resolved in the maintenance phase outlined in chapter 5.2.

5.6.3 Compass

The compass activity can be launched by clicking the live compass icon in the bottom-left area of the map. The compass uses the sensor hardware on the device it is running on to listen for sensor events and adjust the direction of the compass accordingly. The distance between device and vehicle is calculated using geographical coordinates. The compass is only visible if the user has parked their vehicle. A TextureView is set as the background layout so that a camera preview can be rendered behind the compass while the user is walking. This is activated by clicking the camera icon on the bottom-right of the screen shown in figure 5.8. This camera preview allows the user to navigate to their vehicle offline, as well as being aware of their surroundings if they are focused on the screen.

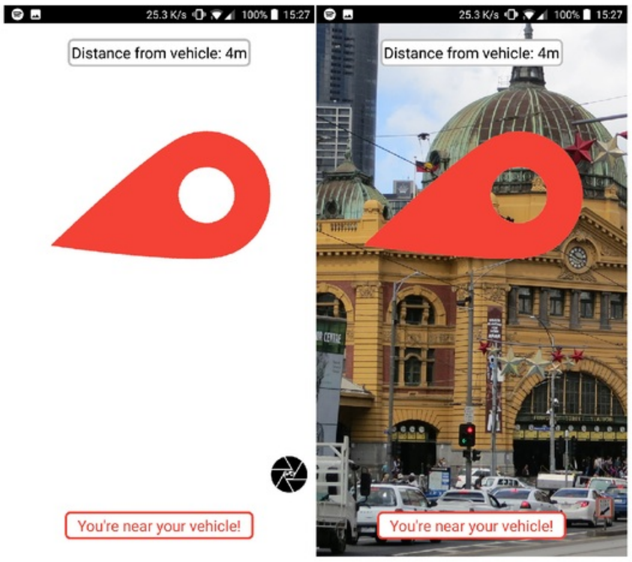

25.3 K/s 100% 15:27
Distance from vehicle: 4m
You're near your vehicle!
25.3 K/s 100% 15:27
Distance from vehicle: 4m
You're near your vehicle!

Figure 5.8 – Screenshots of the compass activity in Little Locator

5.6.4 WHERE

WHERE is the name of the feature that is used to hypothesise where a user has parked based multiple sets of analysed data. There are two main areas in the execution of this feature. The first is a location service that is elected to run in the background. The service keeps track of the location, speed of a user throughout time and stores information on when it is possible that the user has parked. The second area is the algorithm that iterates through all the collected data to give the user the best results for finding their vehicle. The data comes from two sources:

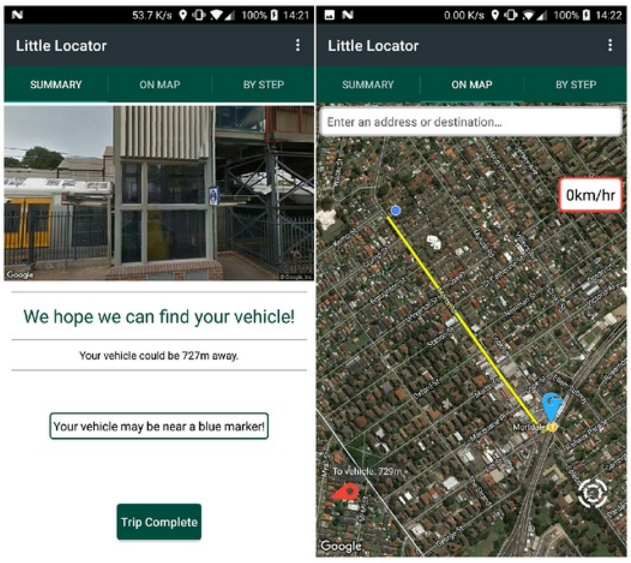

Little Locator
SUMMARY
ON MAP
BY STEP
We hope we can find your vehicle!
Your vehicle could be 727m away.
Your vehicle may be near a blue marker!
Trip Complete
Enter an address or destination...
0km/hr
To vehicle: 729m

Figure 5.9 – Screenshots showing the functionality of WHERE

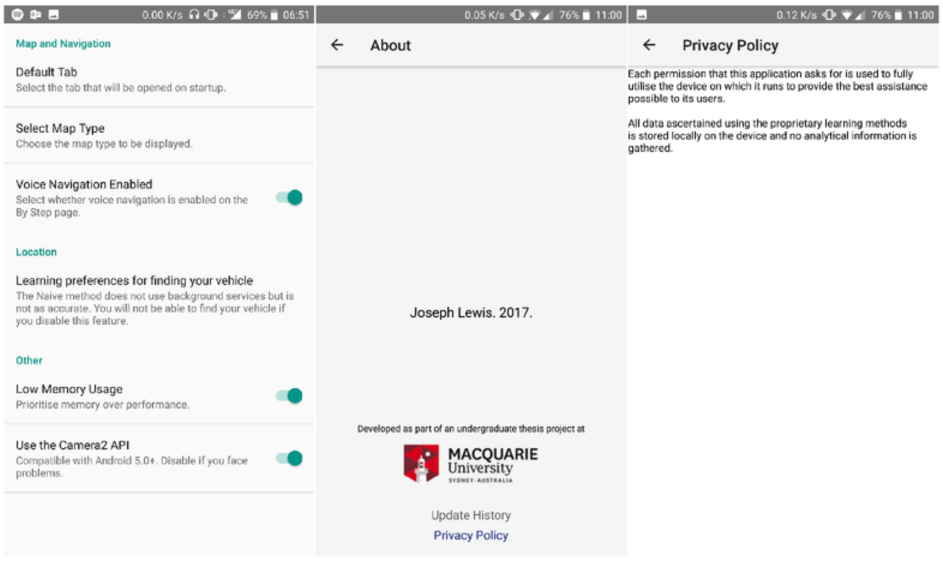
- Every time a user parks their car, the location is made persistent for retrieval if a user needs to find their vehicle in proximity of a previous park. This is due to people generally parking in similar locations when they revisit an area. This information is stored when the learning type is set to anything but off in the application settings.
- When the location service is enabled through "Advanced Learning" in the application settings, the location service marks locations with additional information such as altitude and a timestamp when the algorithm assumes a user has parked.

The location service has been tuned to run execute with shorter restart intervals when a user is driving and at longer intervals when a user is at walking speed or sedentary to conserve battery. When it is assumed that the user is driving, changes in speed are analysed and the weight of a variable called wasDrivingNowParkedFor_WEIGHT is incremented as the algorithm becomes surer that the user has parked over time. Once the weight is high enough the information at that location is stored in a custom location model and placed in a SQLite database instance. The algorithm is more complex than this simple explanation, as the algorithm has been tuned over months to know what pattern last X amount of speeds should follow when a vehicle has been parked, and do its best to ensure that the vehicle is not at a red light. A contributing factor to this cognisance is an opposing variable called reset_WEIGHT that resets wasDrivingNowParkedFor when it is sure enough that the user has not parked. The code extract below demonstrates this. If the weight of the parking assumption is not increased it is assumed the inverse and the reset weight is incremented.

```
...
} else {
    if (wasDrivingNowParkedFor_WEIGHT > 1) {
    reset_WEIGHT += 1;
    ...
```

When the service automatically estimates that a vehicle has been parked a notification prompts the user to manually park their car if application as long as notifications are enabled. There is a large focus on user privacy in this feature as sensitive data is collected. All data is stored locally in SharedPreferences or the SQLite database instance, and can be wiped from the application settings at any time. Another advantage of this implementation is that while it allows more accurate results, an active network connection is not required for the location service to run. When the WHERE button on the default launch screen is clicked the algorithm will process all the information obtained to display the number of calculated estimates on the ma, with the most probable location set with a blue marker image. Figure 5.9 shows a single estimate for the location of the user's vehicle. When clicked, the human readable address is shown, which would be Mortdale Station, NSW.

5.6.5 Miscellaneous


Map and Navigation

Default Tab
Select the tab that will be opened on startup.

Select Map Type
Choose the map type to be displayed.

Voice Navigation Enabled
Select whether voice navigation is enabled on the By Step page.

Location

Learning preferences for finding your vehicle
The Naive method does not use background services but is not as accurate. You will not be able to find your vehicle if you disable this feature.

Other

Low Memory Usage
Prioritise memory over performance.

Use the Camera2 API
Compatible with Android 5.0+. Disable if you face problems.

About

Joseph Lewis. 2017.

Developed as part of an undergraduate thesis project at

MACQUARIE University SYDNEY·AUSTRALIA

Update History
Privacy Policy

Privacy Policy

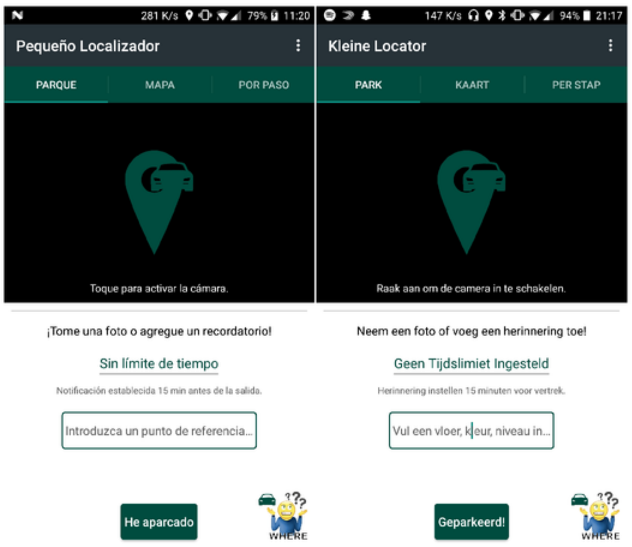
Each permission that this application asks for is used to fully utilise the device on which it runs to provide the best assistance possible to its users.

All data ascertained using the proprietary learning methods is stored locally on the device and no analytical information is gathered.

Figure 5.10 – Screenshots of the settings layout (left), about screen (centre) and by privacy policy (right)

Some additional features that do not warrant their own subsection are listed in this part of the section. The settings, about screen and privacy policy are components that provide extra functionality or information relating to the application. The settings screen allows application functionality to be set, including what tab the application should open to on launch, the map type, and voice navigation settings. In the "Other" section the low memory usage option simply limits the number of background tabs loaded to save memory. The user also has the option to use either version 1 or 2 of the hardware camera API if their device supports it. This is because one API could work better than another on a specific device.

Localisations for Dutch (NL) and Spanish (ES) have been integrated into the application as xml files.



Pequeño Localizador
PARQUE
MAPA
POR PASO
Toque para activar la cámara.
¡Tome una foto o agregue un recordatorio!
Sin límite de tiempo
Notificación establecida 15 min antes de la salida.
Introduzca un punto de referencia...
He aparcado
WHERE
Kleine Locator
PARK
KAART
PER STAP
Raak aan om de camera in te schakelen.
Neem een foto of voeg een herinnering toe!
Geen Tijdslimiet Ingesteld
Herinnering instellen 15 minuten voor vertrek.
Vul een vloer, kleur, niveau in...
Geparkeerd!
WHERE

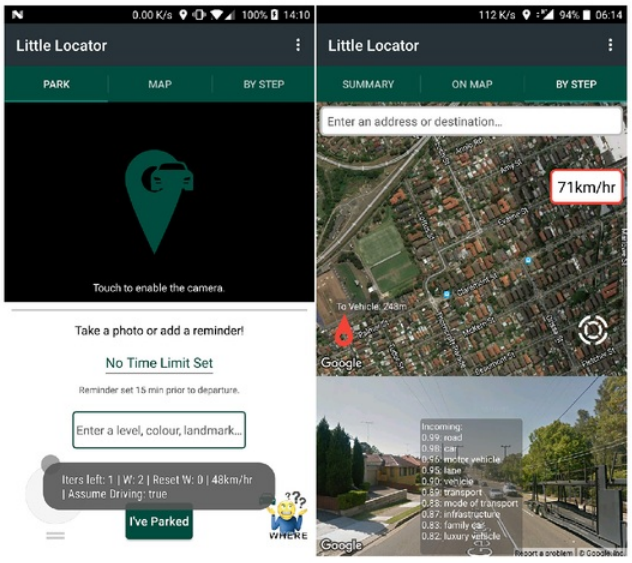
Figure 5.11 – Screens showing ES (left) and NL (right) localisations

5.7 Testing

QA, and testing were large parts of the project. A focus on usability was deemed almost as important as the reliability and accuracy of the application's functionality as one of the objects pertained to a focus on usability, on top of the survey feedback stating that participants wanted a focus on simplicity.

Testing was kept to a close circle with the only author and a single non-technical individual, who wishes to be identified as Amanda, completed QA throughout prototype revisions discussed in chapter 5.8. The focus on this form of testing was to ensure a fluid and effective user experience. The rest of the testing was completed solely by the author.

Code testing comprised of observing outcomes, and creating toasts, logs and noting observations as testing aids. Viewing the revision history in appendix B it can be seen that many stories were related to fixes bugs. A large proportion of time was spend preventing FCs from occurring. This meant implementing error handling through specialised control flows and try-catch blocks for different scenarios. Most of the errors arose from NPEs that occurred if the network or GPS service was disabled, or due to erroneous states when accessing data present while asynchronous tasks were still running.



Little Locator
PARK
MAP
BY STEP
Touch to enable the camera.
Take a photo or add a reminder!
No Time Limit Set
Reminder set 15 min prior to departure.
Enter a level, colour, landmark...
I've Parked
WHERE
Little Locator
SUMMARY
ON MAP
BY STEP
Enter an address or destination...
71km/hr
To Vehicle: 248m
Google
Google

```
Iters left: 1 | W: 2 | Reset W: 0 | 48km/hr
| Assume Driving: true
```

```
Incoming:
0.99: road
0.98: car
0.96: motor vehicle
0.95: lane
0.90: vehicle
0.89: transport
0.88: mode of transport
0.87: infrastructure
0.83: family car
0.82: luxury vehicle
```

Figure 5.12 – Temporary toasts and text to show additional information for application testing

The third primary area of testing was with the navigation and WHERE features. The tests conducted for this functionality relied on observational data being collected and compared to the result in the application. This included:

- Comparing distances between points and estimated walking times in the application to the output of Google Maps and Nokia HERE Maps with the same coordinates,
- Comparing a vehicle's speedometer reading to the speed reading in the application,
- Practical use of the WHERE feature, comparing the estimates to the real scenarios, and
- Testing the accuracy of the computer vision.

The results for the quantitative test data obtained will be discussed in chapter 6.

5.8 Prototype Refinement

After each merge to the master branch it could be considered that there were numerous working prototypes, however, there was a focus on four main prototypes up to the current point in time.

5.8.1 Prototype 1

Version 0.12 – 18/5/17

The initial prototype was ready by the time the project plan was submitted in the first semester of 2017. This prototype contained the basic framework for the application, sans the computer vision assisted navigation and WHERE.

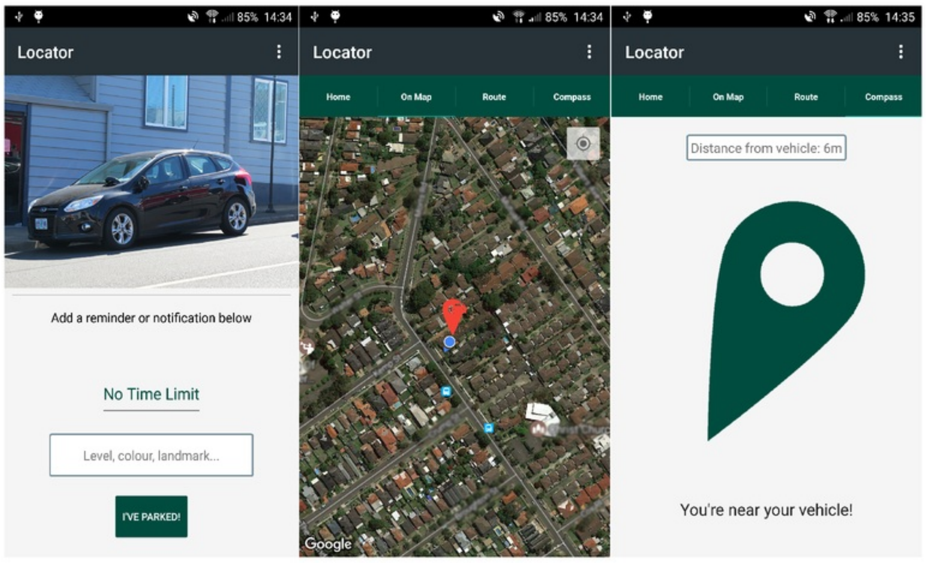
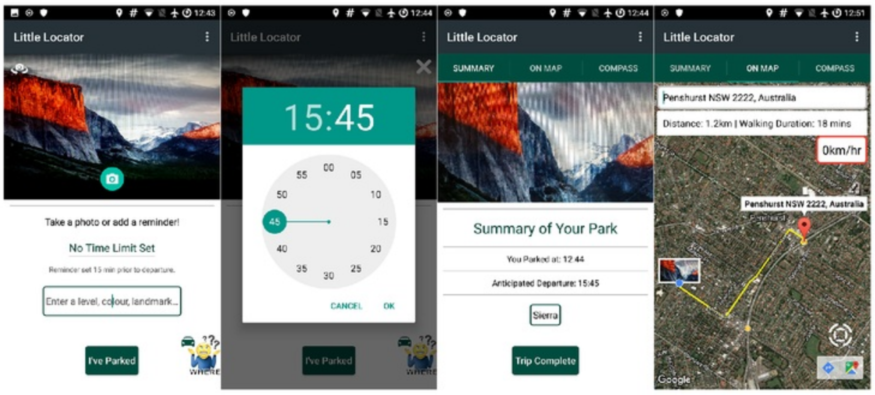
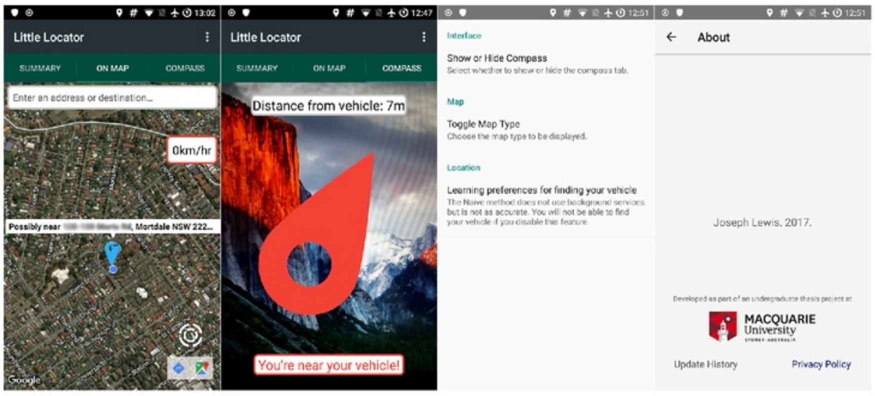

Locator
Home
On Map
Route
Compass
Add a reminder or notification below
No Time Limit
Level, colour, landmark...
I'VE PARKED!
Distance from vehicle: 6m
You're near your vehicle!
Google

Figure 5.13 – Screenshots from the prototype at the time of the project plan submission

5.8.2 Prototype 2

Version 0.15 – 27/07/2017


Little Locator
Take a photo or add a reminder!
No Time Limit Set
Enter a level, colour, landmark...
I've Parked
15:45
CANCEL
OK
SUMMARY
ON MAP
COMPASS
Summary of Your Park
You Parked at: 12:44
Anticipated Departure: 15:45
Sierra
Trip Complete
Penshurst NSW 2222, Australia
Distance: 1.2km | Walking Duration: 18 mins
0km/hr
Google


Little Locator
SUMMARY
ON MAP
COMPASS
Enter an address or destination...
0km/hr
Possibly near [redacted], Mortdale NSW 222...
Little Locator
SUMMARY
ON MAP
COMPASS
Distance from vehicle: 7m
You're near your vehicle!
About
Joseph Lewis. 2017.
Developed as part of an undergraduate thesis project at
MACQUARIE University
Update History
Privacy Policy

Interface

Show or Hide Compass
Select whether to show or hide the compass tab.

Map

Toggle Map Type
Choose the map type to be displayed.

Location

Learning preferences for finding your vehicle
The Naive method does not use background services but is not as accurate. You will not be able to find your vehicle if you disable this feature.

Figure 5.14 – Screenshots from the prototype at the time of the ANZSCON 2017 demonstration

A prototype was required for demonstrative purposes when presenting this project at ANZSCON 2017. This prototype contained a machine learning algorithm based on unsupervised learning with K-means clustering, but still did not contain integration with Google Vision API for navigation. The functionality of WHERE in this prototype was deemed a suitable proof of concept, but ineffective in its implementation as the algorithm took too long to provide effective estimates, was resource heavy and provided many false positives. In-depth detail of this implementation will be omitted as this solution has been completely abandoned. The reason that the preference to enable the location service in the application settings is titled "Learning preferences..." is due to the initial implementation. This naming convention has been retained as a new implementation of machine learning is planned to be added in future revisions, discussed in chapter 9. Additional aesthetic changes were made such as layout object positioning and updated images resources.

5.8.3 Prototype 3

0.50 – 22/09/2017

The third prototype was ready when computer vision assisted navigation and TTS functionality were implemented. In this prototype, the machine learning algorithm had also been replaced with the more effective current location service implementation. The release of this prototype signified the start of the comprehensive testing phase where the accuracy of the complete feature list was scoped.

5.8.4 Prototype 4

0.96 – 05/11/2017

The prototype up to the point that the final project was due. The application is still considered a prototype as several issues are to be implemented before a release would be considered, in addition to the matter of licencing. The majority of the functionality is shippable and this is likely to be the final prototype.

5.9 Licencing

As this is a research project, there was not a significant focus on commercialisation. This aspect of the application will be decided in future, after a discussion with Dr Robert Abbas.

There is one instance of open source code used. The code was adapted from GDirectionsApiUtils [17] and was used for creating a polyline route from parsed a JSON object returned from the Google Maps Direction REST API for navigation. There was no explicit licence included, however, the proper legal and ethical procedures should be following in this matter. This matter would be resolved before release by either proper accreditation of the author and open-sourcing the applicable code, or re-writing the function with completely proprietary code. The class where this code was utilised is called RouteJSONParser and is viewable in appendix D.

Chapter 6

Results

6.1 Requirements Validation

Table 6.1 contains a summary of the features implemented with a description of the state of the functionality, reliability and usability of that item. Table 6.2 outlines the results of additional measures for performance concerning the application.

Table 6.1 – Results from feature usage in Little Locator

Feature	Functionality	Reliability	Usability
Saving and retrieving vehicle locations	Fully implemented with additional functionality	The implementation is reliable. Accuracy depends on the user's device	Refined interface design, utilising multi-threading to reduce lag
User interface	The interface allows access to all implemented features	The interface allows all currently implemented features to be effectively accessed	The UI has been refined to ensure optimal usability through QA
Geocoding	Fully working geocoding and reverse geocoding	Two independent services are being used for reverse Geocoding, Android and Google. The functionality is as reliable as the providers	Enhances usability and is automatically implemented so the user does not have to do anything
Offline GPS compass	Fully implemented	The implementation is	The additional

		reliable. Accuracy depends on the user's device	functionality previously outlined enhances usability
Routing	Routing has been implemented, including computer vision assisted navigation and TTS	The implementation is reliable, however more optimisations to routing are in the project backlog	Modelled with common mapping elements included to encourage user acceptance. Usable based on the requirements of the UI
WHERE	Functional	Reliable, especially compared to available solutions. Google's respective feature is known to regularly provide false positives [9]. The results from the testing of this feature are displayed in chapter 6.1.1	The feature can be activated with a single click of a button. The activation mode can be easily changed from the preference activity
Preferences	Every preference is functional	Preferences are reliably stored and retrieved from the SharedPreferences of the device the application is installed on	The preferences follow the Android development and design guidelines

Table 6.2 – Results of additional application measures

Additional Measure	Result
Stability	The application is stable. All errors that have been found have either had their causes fixed or handlers were added.
Performance	Since revision 0.20 of Little Locator performance has drastically improved due to the utilisation of multithreading for all intensive features. The application is fluid and performs

	tasks efficiently. The only noticeable performance limitation is with the use of computer vision assisted navigation where a sufficient internet connection is required.

6.2 Observation Results for WHERE

Table 6.3 contains amalgamated results of the testing carried out during multiple revisions. All these results are based of observational data. The left column specifies the average time taken for the algorithm to correctly notify the user that it thinks they have parked. The next column shows the average number of false positive when driving on a trip. These could occur due to poor algorithmic structure or being tricked by a red light. The next column shows the percentage that a guess was correctly place when the vehicle was parked, and the final column displays additional notes related to that revision. The sample sizes for each version were not identical, but over ten each to provide sufficient results from their averages.

Table 6.3 – Tabularised results of testing the accuracy of WHERE

Version Code	Time from park until correct guess (s) (avg)	False positives / 5 min (avg)	Correctness of final estimate %	Notes
0.15	30	8	33	Ineffective machine learning implementation
0.50	20	4	60	Early revision of the current implementation
0.80	18	3	73	Minor improvements made
0.90	6	12	6	A code block was incorrectly moved which resulted in the incorrect storage of values
0.94	12	1	94	The code from 0.90 was fixed and values were tweaked

The current implementation of WHERE has been shown to be effective in practical usage, although false positives can still occur. An additional service using a similar K-means clustering algorithm is planned to be implemented as future work. The feature is discussed in chapter 9 and can be trained to identity if the current event is a traffic light, as well as identifying the mode of transportation so that notifications will not appear when the user is catching a train or bus.

Chapter 7

Discussion

7.1 Challenges

Time restraints have been one of the most challenging aspects of the project. Although students have been given the majority of the year to complete their projects, creating a solution of this scale has been difficult in a team of one with other commitments. The difficulty arose from attempting to implement numerous features while simultaneously testing and refining the current features. Nevertheless, a solution to the research problem has been implemented and the current challenges relate to adding value to the product by implementing the non-goals elucidated for the project.

The most technically challenging aspect of the project to date has been optimising the location service class used for WHERE. The difficulties arose from ensuring accurate estimates while also ensuring efficiency. If the location of the device was accessed constantly, the battery drain on the device would be critical, while polling the device too scarcely would produce inaccurate results. The majority of the most recent time allocated to the project was spent refining the feature.

Implementation of effective navigation has been the next challenge that faced in the project. Due to time restraints, a sufficient amount of time could not have been spent on computer vision assisted navigation. The images cannot currently be analysed with depth perception so warnings can be inaccurate. Ideas for improvement are discussed in chapter 9.

7.2 Expected Outcomes

This application can be utilised by any individual with an Android device if it is published on the Google Play Store. Once the remaining major issues have been resolved and licencing is organised, the application has a strong chance of superseding products in the same market segment. It can be expected that once the application is trending, unique downloads will be in the high thousands, based on the research conveyed earlier in the document. If maintenance and development continues, this number will rise and the application can be integrated into peoples' daily lives to more conveniently find their parked cars and aid visually impaired users in navigating their environment. Information stored via the application can also be synced with family members or friends using the application or a navigation app to easily navigate to where the owner's vehicle is located when meeting up or borrowing the vehicle. The expected outcomes, performance measures, means of verification and tangibility are considered in Table 7.1. This hypothesis is supported by the data obtained through the survey conducted earlier in the project.

Table 7.1 – Project Outcomes

Outcome	Performance Measures	Means of Verification	Tangibility
Allow people to effectively relocate their parked vehicle efficiently	- Less time used in returning to vehicles - Less uncertainty when locating vehicles	- Google Play Store reviews - User feedback	No tangibility for this outcome
Automatically estimate where users have parked their vehicles using gathered data	- Added value to the solution	- Google Play Store reviews - User feedback	No tangibility for this outcome
Enhanced navigation for the visually impaired	- Added value to the solution - Usability for visually impaired individuals	- Google Play Store reviews - User feedback	No tangibility for this outcome
Increase the number of users of the application	- Noticeable increase in users	- Number of downloads in the Google Play Store	While this is favourable, it is not crucial to answering the research question.

		- User feedback	Reaching this outcome would show that the solution is effective
Facilitate navigation for visually impaired users	- Utilisation and feedback from the demographic	- Google Play Store reviews - Application analytics	This is a short-term goal and only tangible if it is found not to be feasible

7.3 Impacts

The solution to the research problem and implementation of the non-goals provides an efficient, user-friendly method of relocating parked vehicles to its users. This application could be used by the elderly and less capable, at sporting and entertainment events, in large parking lots and unfamiliar areas. The result would be reduced delays in returning to vehicles and minimising congestion in parking lots at busy times.

An accurate implementation of automatically estimating where users parked is a key feature that has not been successfully publically released by any entity apart from Google up to the time of the research conducted. The impact of this feature would be that users can simply utilise this feature without having to concern themselves with Google's privacy policies. Google tracks and stores user behaviour, location, voice, input and browsing data to provide their services, but also to sell to third-parties as stated in their privacy policy. This application is optimised and completes its computations locally so there are no privacy concerns for users. This feature is also a key development focus and will have more support than Google's similar service.

Functionality to provide navigation with a greater detail of obstructions and the environment, coupled with text-to-speech options is innovative and would make it easier for the visually impaired to navigate independently. During the literature review conducted in chapter 2, there has been no work on assisted navigation through image analysis for smartphones. This feature would be the first of its kind, hence having an edge in the market.

The impacts this solution would have on society would not be too large; although utilisation of this solution would increase productivity in the individuals who utilise this application and provide extra convenience to society. This is especially true for those with poorer memory retentions, or friends, family and co-workers who need to locate a vehicle of another user who parked their vehicle.

Chapter 8

Conclusions

The primary objective of this project was to research and develop and application for Android smartphones that could save the location of a user's parked vehicle for relocation and include a feature that notifies them of when they must depart. This object has been successfully implemented and validated through testing and QA. Due to a large part of the project being related to research, additional requirements for non-goals were elucidated to ensure the solution was the best that could be provided in the scope of the problem and in the allocated timeframe.

It is evident that once refined, computer vision assisted navigation could be a significant new feature in mapping technology. Although this feature is in an early stage of development, through feedback from individuals known to the author and academics at ANZSCON 2017 it has been shown that people have an interest in this area and can see its value. Similarly, the ability of the application to automatically pinpoint the location in which a vehicle was parked locally and without a network connection is a feature that may have significance in the market of smartphone applications. This functionality could be extended to other areas such as tracking technology.

The multiple areas of research were effective in identifying the needs of the public and the technology available to make parked vehicle relocation and navigation more efficient and accessible for everyone. With this research, a solution that supersedes other products available at the current time in several key areas has been developed with a trajectory for future work in this category of applications.

Chapter 9

Future Work

9.1 Future of the Application

Some future work is needed before the application is shippable. The items that must be completed are listed on the issues page of the Bitbucket repository in which the application is located.

The highest priority issues include:

- Improving image analyse to provide more accurate information on where the user has parked when a photo has been taken of the environment. Integration of artificial depth perception is a key goal in improving this form of navigation.
- Implementing a proprietary offline machine learning algorithm using K-means clustering to determine the mode of transportation that a user is taking. This could not only be used to save device resources when the service is not needed and provide more accurate results, but also add value to society in other application if this implementation was made open source. The reason that this method of unsupervised learning has been chosen is that K-means clustering is an effective way of sorting sensor measurements [17].
- Targeting Android API level 27. The project currently targets Android API level 22 as during the early stages of development the author was unsure what permissions were needed and it was more time-efficient to not deal with handling permission access required in Android API level 23 and over. Nonetheless, the application has kept up with other current Android design principles, such as updating depreciated APIS and implementing an ongoing notification to allow the location service to be activated multiple times as of Android 8.0.

9.2 Future Research

Future research is also required in this area, particularly with implementing a light-weight and efficient algorithm for learning transportation modes. Furthermore, research and surveys concerning what features would benefit the visually impaired and less capable should be conducted to ensure that these types of applications benefit those who need them most. From that point, requirements and application designs can be optimised to cater for these individuals, creating an accessible way for navigation and giving more people their independence back.

9.3 Final Words

Accessibility functionality is important for a technologically inclusive society. A focus on creating features that will benefit people who need extra support such as automatic parked vehicle relocation, enhanced navigation and innovative new ideas is important for moving forward in future research.

Chapter 10

Abbreviations

ANZSCON	Australia and New Zealand Student and Young Professionals Congress
API	Application Programming Interface
EULA	End-User Licence Agreement
FC	Force Close
FDD	Feature-Driven Development
GPS	Global Positioning System
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
NPE	Null Pointer Exception
QA	Quality Assurance
TTS	Text-to-Speech
UI	User Interface

References

[1] Google. 2017. Top charts. (August, 2017). Retrieved 16 August, 2017 from https://play.google.com/store/apps/top.

[2] Apple. 2017. iTunes charts. (August, 2017). Retrieved 16 August, 2017 from https://www.apple.com/au/itunes/charts/.

[3] Cory McNutt. 2014. Featured: Top 10 Best Android Car Finder/Parking Apps. (March, 2017). Retrieved 27 May, 2017 from https://www.androidheadlines.com/2014/03/featured-top-10-best-android-car-finderparking-apps.html.

[4] Sam Coch. 2017. 7 Best iPhone Parking App Locate Public Parking Lot & Parked Car Finder. (March, 2017) Retrieved 27 May, 2017 from http://mashtips.com/ios-car-locator-parking-space-finder-apps/.

[5] George Tinari. 2016. Top 3 iOS Apps to Remember Where You Parked Your Car. (March, 2016). Retrieved 27 May, 2017 from http://www.guidingtech.com/56948/top-ios-apps-remember-parked-car/.

[6] DataWRX. 2017. Car Compass. (February, 2017). Retrieved 27 May, 2017 from https://play.google.com/store/apps/details?id=com.datawrx.android.carcompass.

[7] Talent Apps. 2017. ParKing Reminder: Find my car. (May, 2017). Retrieved 27 May, 2017 from https://play.google.com/store/apps/details?id=il.talent.parking.

[8] Talent Apps. 2017. ParKing Reminder: Find my car. (August, 2017). Retrieved 31 August, 2017 from https://play.google.com/store/apps/details?id=il.talent.parking.premium&hl=en

[9] Jessica Dolcourt. 2014. How to find your car with Google Now. (June, 2014). Retrieved 27 May, 2017 from https://www.cnet.com/au/how-to/how-to-find-your-car-with-google-now-parking-locator-card/.

[10] Nick Vega. 2017. Google Maps Can Now Find Your Parked Car For You. (March, 2017). Retrieved 31 August, 2017 from https://www.lifehacker.com.au/2017/03/google-maps-can-now-find-your-parked-car-for-you/.

[11] Lance Whitney. 2016. Apple's updated Maps app will recall where you parked the car. (June, 2016). Retrieved 31 August, 2017 from https://www.cnet.com/news/apples-new-maps-app-will-remember-where-you-parked-your-car/.

[12] Theredsunrise. 2015. Instago Street View Navigation. (May, 2015). Retrieved 17 May, 2017 from https://play.google.com/store/apps/details?id=com.theredsunrise.whereami&hl=en.

[13] Mohit Gupt. 2015. Android Tabs Example – With Fragments and ViewPager. (June, 2015). Retrieved 29 May, 2017 from http://www.truiton.com/2015/06/android-tabs-example-fragments-viewpager/.

[14] Apple. 2017. Navigation Bars - UI Bars - iOS Human Interface Guidelines (May, 2017). Retrieved 29 May, 2017 from https://developer.apple.com/ios/human-interface-guidelines/ui-bars/navigation-bars/.

[15] IDC. 2017. Smartphone OS Market Share, 2017 Q1 (May, 2017). Retrieved 14 October, 2017 from https://www.idc.com/promo/smartphone-market-share/os.

[16] Zain. 2017. Understanding the Agile Software Development Lifecycle and Process Workflow (January, 2017). Retrieved 30 May, 2017 from https://www.smartsheet.com/understanding-agile-software-development-lifecycle-and-process-workflow.

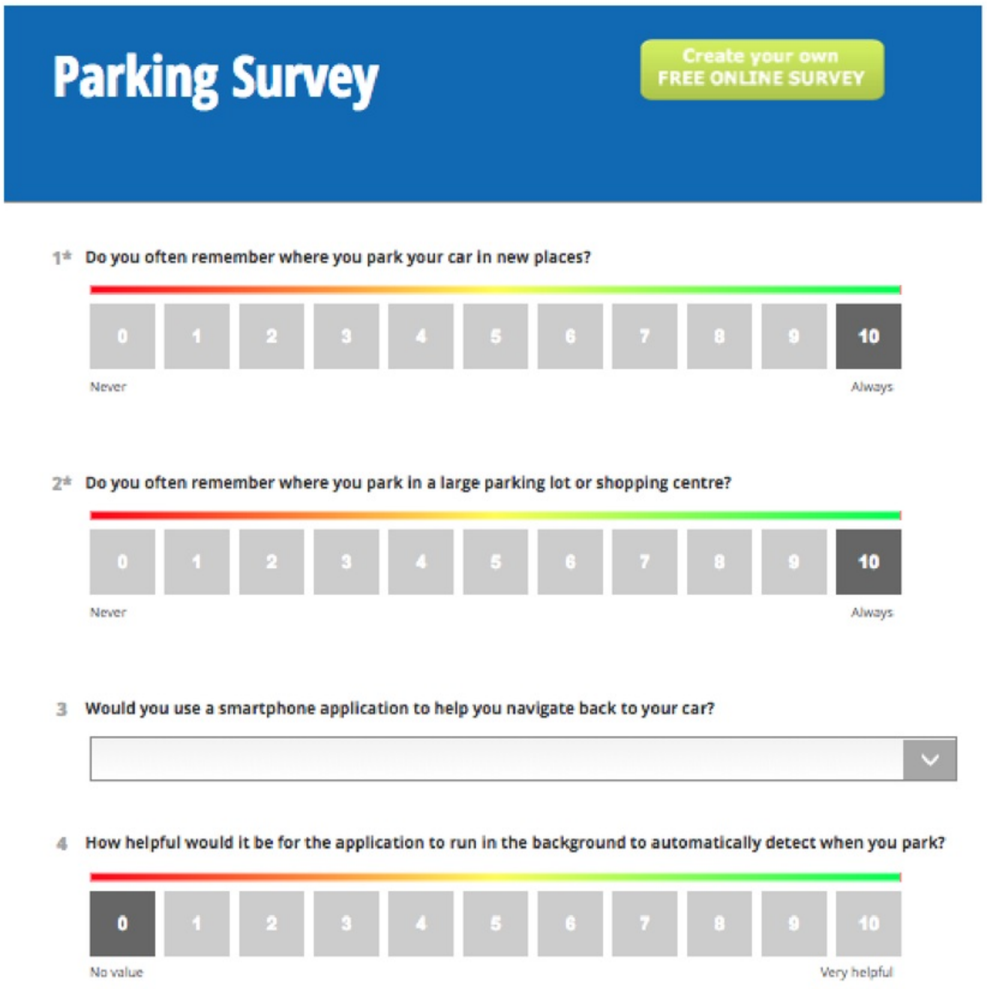
[17] Mathias Seguy. 2014. GDirectionsApiUtils (November, 2014). Retreiced 11 July, 2017 from https://github.com/MathiasSeguy-Android2EE/GDirectionsApiUtils/blob/master/GDirectionsApiUtils/src/com/an

droid2ee/formation/librairies/google/map/utils/direction/parser/DirectionsJSON Parser.java.

[18] Andrea Trevino. 2016. Introduction to K-means Clustering (June, 2016). Retrieved 16 October, 2017 rom https://www.datascience.com/blog/k-means-clustering.

Appendix A

Parking Survey



Parking Survey

1* Do you often remember where you park your car in new places?

0 1 2 3 4 5 6 7 8 9 [✓ 10]

Never — Always

2* Do you often remember where you park in a large parking lot or shopping centre?

0 1 2 3 4 5 6 7 8 9 [✓ 10]

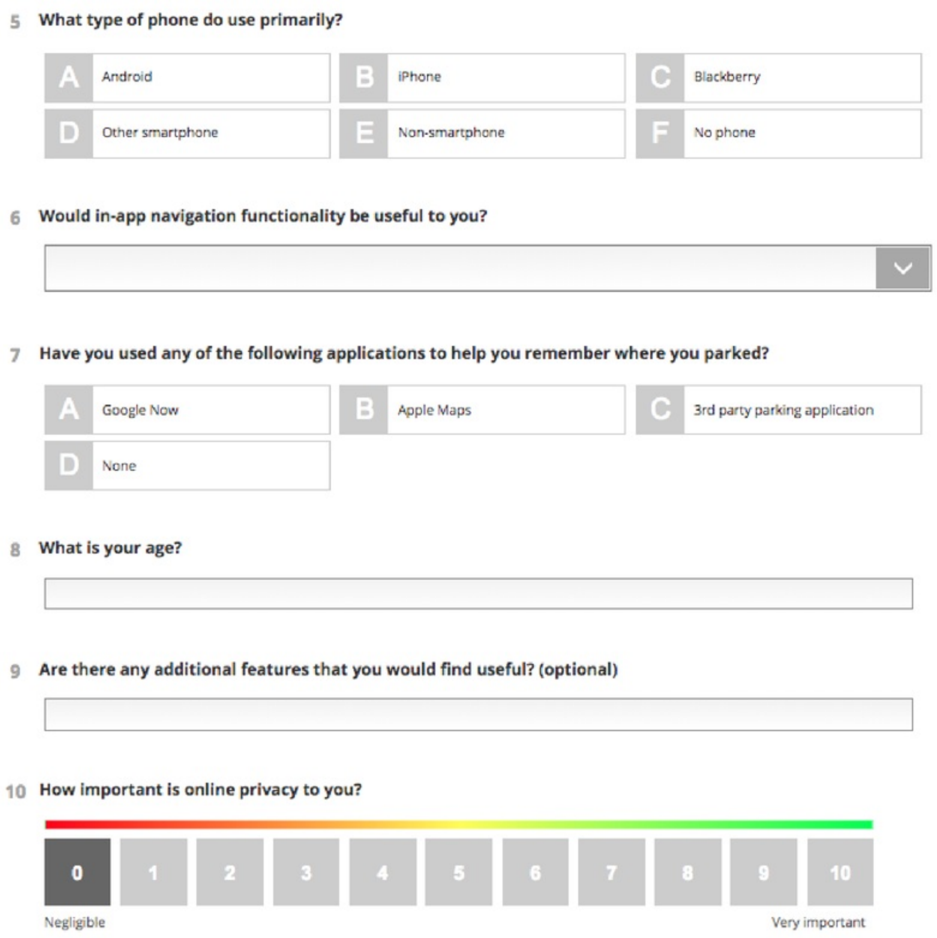
Never — Always

3 Would you use a smartphone application to help you navigate back to your car?

4 How helpful would it be for the application to run in the background to automatically detect when you park?

[✓ 0] 1 2 3 4 5 6 7 8 9 10

No value — Very helpful



5 **What type of phone do use primarily?**

A Android	B iPhone	C Blackberry
D Other smartphone	E Non-smartphone	F No phone

6 **Would in-app navigation functionality be useful to you?**

7 **Have you used any of the following applications to help you remember where you parked?**

A Google Now	B Apple Maps	C 3rd party parking application
D None		

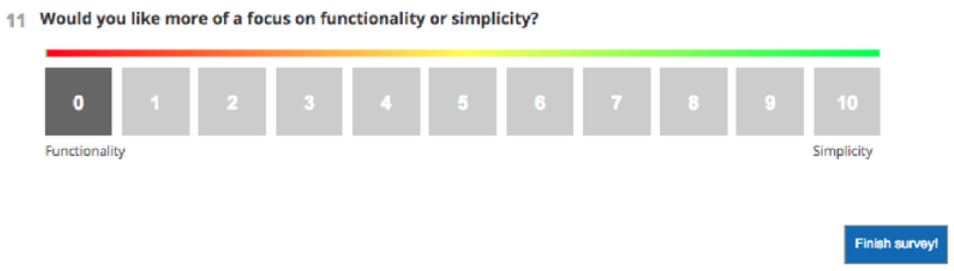
8 **What is your age?**

9 **Are there any additional features that you would find useful? (optional)**

10 **How important is online privacy to you?**

[✓ 0] 1 2 3 4 5 6 7 8 9 10

Negligible … Very important


11 Would you like more of a focus on functionality or simplicity?

Appendix B

Little Locator Revision History

B.1 Overview

This section contains the notes of features, improvements or bug fixes that were implemented with each revision of Little Locator.

B.2 Revision History

0.1 (7/4/17)

- added all fragments

0.11 (18/4/17)

- camera activity replaces intent
- parameterisation, code clean-up

0.12 (18/5/17)

- custom parking pin with photo
- Added basic routing
- UI overhaul
- bitmap caching, removed android:largeHeap="true" (Improve performance)

0.13 (17/6/17)

- Threading for bitmap operation

- Save picture (path) in SharedPreferences, caching

- Bug fixed leading to multiple FC

- fix fc on park if bitmap isn't ready

- Work on notification logic

0.14 (3/7/17)

- handling location and data settings

- Implemented alternative geocoding method due to an android bug

- fix android orientation bug

- improved routing

0.15 (24/7/17)

- improved routing, new primary method, distance and duration, addresses on markers

- Acknowledge MQU in app

- moved strings to strings xml

0.16 (1/8/17)

- added guess where parked

- no distance or duration if route not found

0.17 (7/8/17)

- added settings activity to replace menu items

- blue icon and different text for guess "possibly near"

- Added NL Localisation

- Removed obsolete code

- Converted Strings to xml

- handled max database size

0.18 (17/8/17)

- merged map and routing fragments

- Use speed in clustering algorithm

- added camera preview to compass (18/8)

- tuned location listener algorithm (21/8)

- fixed incorrect decimal offset in speedometer (22/8)

- added GPS fault tolerance for speed (22/8)

0.19 (24/8/17)

- Begin implementing guess time parked

- Shows progress when completing time-consuming tasks (eg. estimating location)

0.20 (25/8/17)

- Utilised threading for map UI updates to greatly improve performance
- Improved progress UI updates
- Modified code to better conform with Android programming practices
- improved clean-up of services on exit
- Animate progress updates in a separate thread
- Store map type in SharedPreferncesc

0.21 (25/8/17)

- background task to keep track of user's movements

0.22 (30/8/17)

- learning type preferences added (26/8)
- optimised services for better battery usage
- Implementation of SQLite db for storage
- bug fixing, stability in control flow (27/8)
- notification settings in SharedPrefernces (28/8)
- fixed preference bugs (28/8)
- updated AppCompat to latest
- updated to latest support libraries, added maven repository support
- fixed negation bug in checking if alarm is on
- updated strings
- start implementation of showing and hiding compass
- location listening in background / optimise so not much draining

0.3, build 14 (6/9/17)

- Implemented a tab supporting street view (5/9)
- Implemented street view configurations to follow user's path (5/9)
- implemented an optimised location getter
- seek user's preferences for learning type on initial use
- fix speedometer [update: test now]
- FIX app crashes on park without data connection
- ensure preferences are persistent
- minor refractor of project in naming to conform to guidelines
- updated to latest Gson library
- fixed string comparison logic error

- updated to the latest GPS - major code update

- UI improvements

0.31, build 15 (7/9/17)

- Added ES localisation

- Format changes

- removed dead code

0.32 build 16 (8/9/17)

- Use new method to get best location for more functionality

0.33 build 17 (10/9/17)

- Added device file clean-up code

- Use street view image in summary if the user did not take a photo - https://developers.google.com/maps/documentation/streetview/intro

- Delete all data from learning if learning disabled

0.40 build 18 (16/9/17)

- added low memory option

- removed hide compass setting

- added more network related error handling

- updated db model

- removed learning from main activity, moved purely to a background service

0.5 build 19 (22/09/2017)

- Added image analysis for navigation

- implemented TTS

- optimisations for Android O (19/9)

- notification opens activity intent

- new method to access compass

- changed preference cases to accommodate default values

- Removed

apache http library, while retaining backwards compatibility

0.51 build 20 (24/09/2017)

- Draw the quickest found route from the user's current location to vehicle upon opening map

- Updated to latest libraries

- Fixed incorrect computer vision label strings

- Fixed TTS memory leak

- Fixed write to parcel if location instance is null

- Resolved httpclient conflict warnings due to clashing libraries

- Consolidated API keys

- Implemented autofill text on double click of the address bar

- Fixed random NPEs

- Optimised API usage

0.52 build 21 (25/09/2017)

- Added voice options setting

- Added more strings and NL, ES translations

0.53 build 22 (02/10/2017)

- Fixed several possible null pointer exceptions causes

- Updated SDK build tools version

- Implemented primitive turn-by-turn navigation with voice

0.60 build 23 (03/10/2017)

- Revamped guessing algorithm made obsolete when SQLite was implemented

- Added navigation text bar (4/10)

- Ensure initial navigation processing (5/10)

- Fixed scoping error (6/10)

- Minor UI element updates

- Prioritise memory setting on by default

- Internal refactoring

0.70 build 24 (07/10/2017)

- Utilise the camera2 API on supported devices

- (Performance) Photos are stored exclusively in byte form, files are no longer generated

0.71 build 25 (08/10/2017)

- Added timestamp to custom location objects from greater accuracy

- Improved location service

- Greatly improved battery usage

- Improved camera logic, code clean-up (9/10)

- Better image sizes and scaling (9/10)

0.80 build 26 (10/10/2017)

- Improved algorithms in the location service and fixed logical errors

- Greatly improved WHERE algorithm

- Removed ML as it added overhead and gave more false positives

- Utilised a quick sort algorithm for data sorting

- Numerous bug fixes

- Added more logging for developmental purposes

- Stop service service when parked

0.81 build 27 (12/10/2017)

- Maps and navigation can be used independently of parking features

- Ensure locations in proximity are not shown redundantly using WHERE

- Don't require internet

- Fixed address not loading occasionally on the Map tab

0.82 build 28 (13/10/2017)

- Updated Google Play Services API

- Fixed NPE when finding a route if not parked

- Choice to keep or clear learnt data on service disabled

- Added choice of custom start location for navigation

- Minor bug fixes

0.83 build 29 (15/10/2017)

- Changed the colour of markers for guessed spots to blue

- Added additional in-app text for the user

- The application is now (limited but) usable without a network connection

- Better handling when location services are disabled or unavailable

0.84 build 30 (23/10/2017)

- Added final network and location service checks before performing specific actions that could force close the application if not handled

- Fixed an issue on load causes a message to appear saying a route could not be found

- Updated the privacy policy in EN, NL and ES

- A live revision history can be downloaded from an external source from within the application

0.85 build 31 (24/10/2017)

- Simplified low memory setting

- Reduced time between background service starts to increase the accuracy of WHERE estimates

0.86 build 32 (26/10/2017)

- Possible location service accuracy improvements

- Upgrading to the latest environment and libraries, reverted Gradle 4.1 as it caused dex problems

0.90 build 33 (28/10/2017)

- Improved location service algorithms

- Improved internal documentation

- Disable the compass when not parked

- Migrated source to Bitbucket

0.91 build 34 (28/10/2017)

- Improved error handling when a location cannot be determined

- Code optimisations

- Corrected some strings

- Removed toasts used for testing and debugging

0.92 build 35 (29/10/2017)

- Update to 26.0.2 SDK compile version

- Fixed navigation points being cleared

- Navigation code clean-up

- Removed redundant code

0.93 build 36 (29/10/2017)

- Improved location service

0.94 build 37 (31/10/2017)

- Improved Location Service

- Improved location setting handling

- Added an ongoing notification when the location service is running - required in Oreo

- Fixed departure time being reset

0.95 build 38 (1/11/2017)

- Logging used for testing purposes has been removed or commented out

- Location service timings have been tweaked for optimal user experience

0.96 build 39 (5/11/2017)

- Updated NL and ES translations

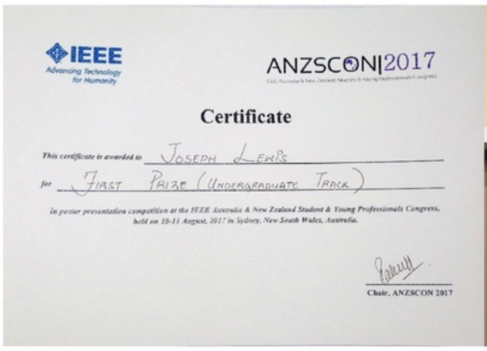
- Updated hyperlinks

- Internal changes

Appendix C

Academia

C.1 ANZSCON 2017



IEEE Advancing Technology for Humanity

ANZSCON|2017

Certificate

This certificate is awarded to JOSEPH LEWIS

for FIRST PRIZE (UNDERGRADUATE TRACK)

in poster presentation competition at the IEEE Australia & New Zealand Student & Young Professionals Congress, held on 10-11 August, 2017 in Sydney, New South Wales, Australia.

[illegible]

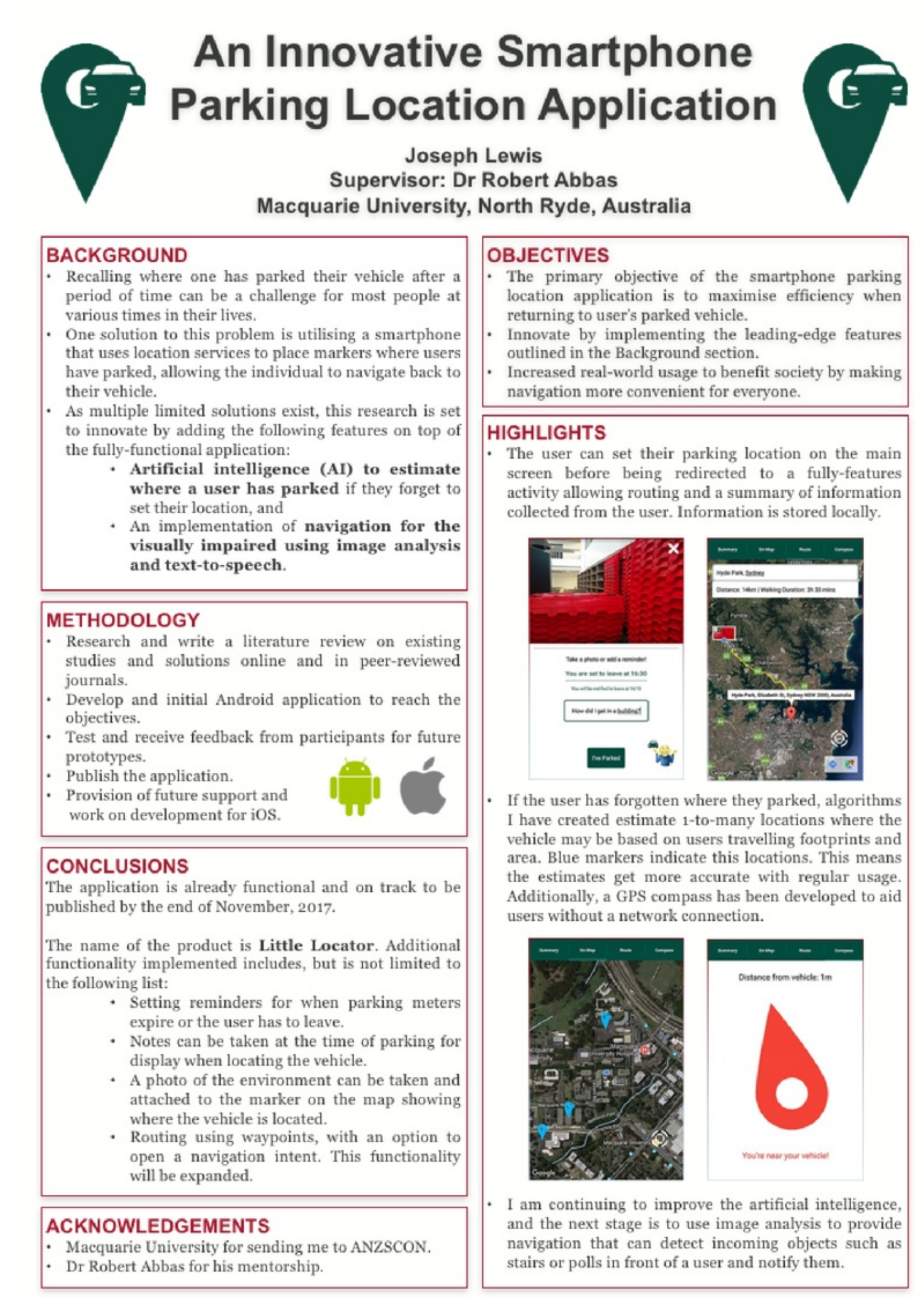
Chair, ANZSCON 2017

Figure 7.1 – Research poster 1st place award (left) and the author receiving the award (right)

On the 10th and 11th of August IEEE ANZSCON 2017 was held in Sydney at UTS. ANZSCON is a congress where students and young professionals from Australia, New Zealand, and neighbouring countries such as Indonesia network and engage with trends in science and technology, develop skills and share experiences. One highlighted agenda at ANZSCON was the poster competition that allowed undergraduate and post graduate students to present their current research in the form of posters and prototypes. The research up to that stage on the application being developed used machine learning and image analysis to make parking and navigation simpler for users. It was an honour to win first place in the undergraduate division at this prestigious event, with the outcome leading to an encouraging experience.

This affirmation supported the hypothesis that the project being developed possessed academic and practical value.

C.2 ANZSCON 2017 Poster Competition Submission



An Innovative Smartphone Parking Location Application

Joseph Lewis
Supervisor: Dr Robert Abbas
Macquarie University, North Ryde, Australia

BACKGROUND

- Recalling where one has parked their vehicle after a period of time can be a challenge for most people at various times in their lives.
- One solution to this problem is utilising a smartphone that uses location services to place markers where users have parked, allowing the individual to navigate back to their vehicle.
- As multiple limited solutions exist, this research is set to innovate by adding the following features on top of the fully-functional application:
    - **Artificial intelligence (AI) to estimate where a user has parked** if they forget to set their location, and
    - An implementation of **navigation for the visually impaired using image analysis and text-to-speech**.

METHODOLOGY

- Research and write a literature review on existing studies and solutions online and in peer-reviewed journals.
- Develop and initial Android application to reach the objectives.
- Test and receive feedback from participants for future prototypes.
- Publish the application.
- Provision of future support and work on development for iOS.

CONCLUSIONS

The application is already functional and on track to be published by the end of November, 2017.

The name of the product is **Little Locator**. Additional functionality implemented includes, but is not limited to the following list:

- Setting reminders for when parking meters expire or the user has to leave.
- Notes can be taken at the time of parking for display when locating the vehicle.
- A photo of the environment can be taken and attached to the marker on the map showing where the vehicle is located.
- Routing using waypoints, with an option to open a navigation intent. This functionality will be expanded.

ACKNOWLEDGEMENTS

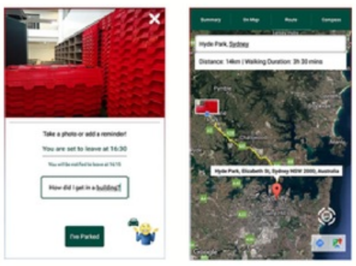
- Macquarie University for sending me to ANZSCON.
- Dr Robert Abbas for his mentorship.

OBJECTIVES

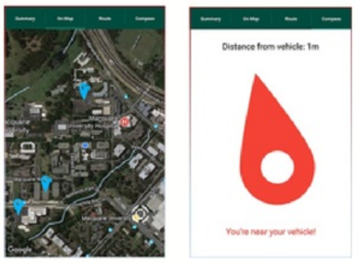
- The primary objective of the smartphone parking location application is to maximise efficiency when returning to user's parked vehicle.
- Innovate by implementing the leading-edge features outlined in the Background section.
- Increased real-world usage to benefit society by making navigation more convenient for everyone.

HIGHLIGHTS

- The user can set their parking location on the main screen before being redirected to a fully-features activity allowing routing and a summary of information collected from the user. Information is stored locally.


Take a photo or add a reminder!
You are set to leave at 16:30
How did I get in a building?
I've Parked
Summary
On Map
Route
Compass
Hyde Park, Sydney
Distance: 14km | Walking Duration: 3h 30 mins
Hyde Park, Elizabeth St, Sydney NSW 2000, Australia

- If the user has forgotten where they parked, algorithms I have created estimate 1-to-many locations where the vehicle may be based on users travelling footprints and area. Blue markers indicate this locations. This means the estimates get more accurate with regular usage. Additionally, a GPS compass has been developed to aid users without a network connection.



- I am continuing to improve the artificial intelligence, and the next stage is to use image analysis to provide navigation that can detect incoming objects such as stairs or polls in front of a user and notify them.

C.3 IEEE Paper

A potential IEEE paper for this project is in the process being finalised, extending the academic pursuit past the previous ANZSCON 2017 event. Publishing this research in an IEEE journal is an important addition to the publishing of this research report in the Macquarie University library as it allows for knowledge to be shared and greater exposure for this research.

Appendix D

RouteJSONParser.java

```
package com.littleinnovations.littlelocator;

import com.google.android.gms.maps.model.LatLng;

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

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 * Created by Joseph Lewis on 11/7/17. (Generated by Android Studio)
 *
 * Adapted from code created by Mathias Seguy:
 *
 * Mathias Seguy. 2014. GDirectionsApiUtils (November, 2014).
 * Retreiced 11 July, 2017 from
 * https://github.com/MathiasSeguy-Android2EE/
 * GDirectionsApiUtils/blob/master/GDirectionsApiUtils
 * /src/com/android2ee/formation/librairies/google/map/
 * utils/direction/parser/DirectionsJSONParser.java.
 *
 * * * IF THIS PROJECT IS TO BE PUBLISHED IT WILL BE
 * ENSURED THAT PROPER
 * LICENCING AGREEMENTS WILL BE ADHERE BY * * *
 *
 *
 * Parses the JSON returned to get a navigation route
 * using polylines.
 */

public class RouteJSONParser {

    /**
     * Returns a route in list form from the parsed JSON object
     *
     * @param jObject the object
     * @return route
     */
    public List<List<HashMap<String, String>>> parse(JSONObject jObject) {

        List<List<HashMap<String, String>>> routes = new
ArrayList<List<HashMap<String, String>>>();
        JSONArray jRoutes = null;
        JSONArray jLegs = null;
        JSONArray jSteps = null;

        try {

            jRoutes = jObject.getJSONArray("routes");


            for (int i = 0; i < jRoutes.length(); i++) {
                jLegs = ((JSONObject) jRoutes.get(i)).getJSONArray("legs");
                List route = new ArrayList<HashMap<String, String>>();


                for (int j = 0; j < jLegs.length(); j++) {
                    jSteps = ((JSONObject)
jLegs.get(j)).getJSONArray("steps");
```

```
                        for (int k = 0; k < jSteps.length(); k++) {
                            String polyline = (String) ((JSONObject)
((JSONObject) jSteps.get(k)).get("polyline")).get("points");
                            List<LatLng> list = decodePoly(polyline);


                            for (int l = 0; l < list.size(); l++) {
                                HashMap<String, String> hm = new
HashMap<String, String>();
                                hm.put("lat",
Double.toString((list.get(l)).latitude));
                                hm.put("lng",
Double.toString((list.get(l)).longitude));
                                route.add(hm);
                            }
                        }
                        routes.add(route);
                    }
                }

            } catch (JSONException e) {
                e.printStackTrace();
            } catch (Exception e) {
            }
            return routes;
        }

        /**
         * Return a list of locations decoded from a string of points
         * @param encoded points
         * @return the list of decoded locations
         */
        private List<LatLng> decodePoly(String encoded) {
```

```
        List<LatLng> polylines = new ArrayList<LatLng>();
        int index = 0, len = encoded.length();
        int lat = 0, lng = 0;

        while (index < len) {
            int b, shift = 0, result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;
                shift += 5;
            } while (b >= 0x20);
            int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >>
1));
            lat += dlat;

            shift = 0;
            result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;
                shift += 5;
            } while (b >= 0x20);
            int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >>
1));
            lng += dlng;

            LatLng poly = new LatLng((((double) lat / 1E5)),
                    (((double) lng / 1E5)));
            polylines.add(poly);
        }
        return polylines;
    }
}
```