User-Configurable, Location-Aware, Portable
UK Map Guide

By
Dana Jon Kamason

A Dissertation
submitted to Heriot-Watt University
in fulfillment of the
dissertation requirement for the degree of

Master of Science in Computing

Heriot-Watt University, Edinburgh, United Kingdom

Supervisor: Dr. Hamish Taylor
August 2012
Non-plagiarism statement

DECLARATION

I, Dana Kamason, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: ...........................................

Date: ..............................................
Abstract

With the increasing attractiveness of mobile computing and mobile communication and the emergence of mobile computers and smartphones with integrated Global Positioning System (GPS) sensors, the demand for location-aware applications - a technology that identifies the physical location of the mobile device in real time - has proliferated rapidly. Modern handheld computers or smartphones with inbuilt GPS sensors invite use as portable location-aware guides in order to aid users to make informed decisions based on geographic knowledge. Also, the widespread deployment of wireless networks and the growing availability of free geographic information via the Internet have facilitated the development of location-aware or geo-location applications.

This dissertation presents a project which is aimed at developing a configurable map guide application for handheld computers or smartphones with an inbuilt GPS sensor, which presents location-sensitive information about any chosen map vicinity in the United Kingdom (UK). This report explains in detail the main technologies that are behind the increased use of location sensitive applications. The application of this project was developed for Android devices using the Eclipse IDE and Android Application Development Tool (ADT) plug-in with the Java programming language. The final result is a configurable, location sensitive UK map guide for portable handheld Android devices.
Acknowledgement

I would like to express deepest gratitude to my project supervisor Dr. Hamish Taylor, for his excellence guidance, patience and shrewd suggesting throughout this project.

Also, I would like to say thank you to all those who participated in this project and contributed in diverse ways.

Finally, special thanks go to my family, for providing continuous encouragement, motivation and support throughout this project.
Table of Contents

Non-plagiarism statement........................................................................................................... i
Abstract....................................................................................................................................... ii
Acknowledgement .................................................................................................................... iii
Table of Contents ......................................................................................................................... iv
List of Figures ............................................................................................................................... vi
List of Tables ............................................................................................................................... vii

CHAPTER 1: INTRODUCTION .................................................................................................. 1
  1.1 AIMS AND OBJECTIVES ................................................................................................. 2
      1.1.1 Portable ....................................................................................................................... 2
      1.1.2 Configurable ............................................................................................................... 2
      1.1.3 Location Awareness ................................................................................................... 3
      1.1.4 User-friendliness ....................................................................................................... 3

CHAPTER 2: LITERATURE REVIEW .................................................................................... 5
  2.1 PROJECT IMPORTANCE ................................................................................................. 5
  2.2 MOBILE COMPUTING .................................................................................................... 6
      2.2.1 GPS Tracking Apps ................................................................................................... 6
      2.2.2 My Tracks .................................................................................................................. 7
  2.3 LOCATION SENSING TECHNOLOGIES ...................................................................... 7
      2.3.1 Global Positioning System (GPS) ............................................................................. 7
      2.3.2 Wireless Network Infrastructure ............................................................................. 9
  2.4 GIS AND DIGITAL MAPPING ........................................................................................ 10
      2.4.1 Geographic Data ....................................................................................................... 11
      2.4.2 Maps .......................................................................................................................... 11
  2.5 DEVELOPMENT TECHNOLOGIES ............................................................................. 15
      2.5.1 Android .................................................................................................................... 15
      2.5.2 Java Programming Language ................................................................................... 18
      2.5.3 Map Data Source ..................................................................................................... 19

CHAPTER 3: REQUIREMENT ANALYSIS ........................................................................... 22
  3.1 FUNCTIONAL REQUIREMENTS .................................................................................... 22
  3.2 NON-FUNCTIONAL REQUIREMENTS .......................................................................... 24
  3.3 PROFESSIONAL, LEGAL AND ETHICAL ISSUES ....................................................... 24

CHAPTER 4: DESIGN AND DEVELOPMENT PROCESS .................................................. 25
  4.1 METHODOLOGY ........................................................................................................... 25
  4.2 APPLICATION DESIGN ................................................................................................. 28
      4.2.1 Location-aware Use Case - Requirements ................................................................. 29
      4.2.2 Configurable Use Case - Requirements ................................................................. 34
  4.3 TWO INCREMENTAL VERSIONS .................................................................................. 38
4.3.1 Increment One ................................................................. 38
4.3.2 Increment Two ............................................................... 44

CHAPTER 5: TESTING ................................................................... 48
5.1 UNIT TESTING .................................................................. 48
5.2 INTEGRATED TESTING ...................................................... 48
5.3 SYSTEM TESTING ............................................................. 50
5.4 USER ACCEPTANCE TESTING .......................................... 52

CHAPTER 6: EVALUATION ........................................................... 53
6.1 EVALUATION APPROACH .................................................. 53
6.2 PARTICIPANTS ................................................................. 53
6.3 QUESTIONNAIRE ............................................................. 55
6.4 EVALUATION OF RESULTS ............................................... 56

CHAPTER 7: CONCLUSION AND RECOMMENDATIONS ............... 62
7.1 CONCLUSION .................................................................. 62
7.2 RECOMMENDATIONS FOR FURTHER WORK ...................... 62

REFERENCES ........................................................................... 64

APPENDIX I: APPLICATION SCREENSHOTS ............................... 66
APPENDIX II: INITIAL PROJECT PLAN ....................................... 71
APPENDIX III: EVALUATION TASKS SHEET AND QUESTIONNAIRE ... 73
APPENDIX IV: GANTT CHART ..................................................... 75
APPENDIX V: CODES (MAIN ACTIVITY) ...................................... 76
List of Figures

Figure 2-1: GPS constellation [www.gps.gov] ......................................................8
Figure 2-2: The three GPS segments [Connet.us] ..................................................8
Figure 2-3: Types of map scales [10] ....................................................................13
Figure 2-4: Developable surfaces in the second stage of map projection [10] .........14
Figure 2-5: Android Systems Architecture - Image taken from [16] ....................17
Figure 2-6: Life Cycle of an Android Activity ........................................................18
Figure 2-7: Google Maps - displayed in a web browser ......................................20
Figure 3-1: Overview of Functional Requirements ............................................22
Figure 4-1: IID methodology – adopted from Adv. SE course note .................26
Figure 4-2: Increment One (version 1.0) use case diagram ............................39
Figure 4-3: Increment One (version 1.0) User Interface design ........................40
Figure 4-4: Increment One Map View ...............................................................42
Figure 4-5: Increment One Satellite View ..........................................................42
Figure 4-6: Graphic view of added location ......................................................43
Figure 4-7: Add details activity ...........................................................................43
Figure 4-8: Added location displayed on the map ............................................43
Figure 4-9: Increment Two (version 1.1) use case diagram ............................45
Figure 4-10: Increment Two (version 1.1) User Interface design ......................46
Figure 5-1: Increment Two Map View ...............................................................49
Figure 5-2: Increment Two Satellite View ..........................................................49
Figure 5-3: Application running on Samsung Galaxy Tab ...............................50
Figure 6-1: Participants' gender [Evaluation analyses] .......................................54
Figure 6-2: Participants' age group [Evaluation analyses] .................................54
Figure 6-3: Participants' skill levels [Evaluation analyses] ..................................55
Figure 6-4: Tasks completion [Evaluation analyses] ..........................................56
Figure 6-5: The application's helpfulness [Evaluation analyses] .......................57
Figure 6-6: Ease of use [Evaluation analyses] .....................................................58
Figure 6-7: User-friendly interface [Evaluation analyses] ..................................58
Figure 6-8: Organisation of menu items [Evaluation analyses] .........................59
Figure 6-9: Smooth running of application [Evaluation analyses] .....................60
List of Tables

Table 3-1: Detailed description of Functional Requirements .............................................. 23
Table 4-1: Get user’s current location - Use Case textual description .............................. 29
Table 4-2: Get x,y coordinates - Use Case textual description ......................................... 30
Table 4-3: Display map in Standard View - Use Case textual description ...................... 30
Table 4-4: Display map in Satellite View - Use Case textual description ....................... 31
Table 4-5: Operate application in GPS mode - Use Case textual description ............... 31
Table 4-6: Operate application in Network mode - Use Case textual description ....... 32
Table 4-7: Calculate Distance from point A to B - Use Case textual description ....... 32
Table 4-8: Calculate Direction from point A to B - Use Case textual description ....... 33
Table 4-9: Register location - Use Case textual description ............................................. 34
Table 4-10: Edit Registered location - Use Case textual description ............................. 35
Table 4-11: Display list of registered locations - Use Case textual description .......... 35
Table 4-12: Zoom In - Use Case textual description ....................................................... 36
Table 4-13: Zoom Out - Use Case textual description .................................................... 36
Table 4-14: Share location information - Use Case textual description ..................... 37
Table 4-15: Version Increments ...................................................................................... 38
Table 5-1: Structured Test Plan ......................................................................................... 51
Chapter 1: Introduction

This project is aimed at developing a configurable map guide for handheld computers or smartphones with an integrated Global Positioning System (GPS) to present location sensitive information about any chosen map vicinity in the United Kingdom (UK). Modern handheld computers or smartphones with inbuilt GPS sensors invite use as portable guides. This application is also aimed at providing users with the ability to locate positions easily. LCD (liquid crystal display) screens on mobile computers can display multi-way scrollable maps in bright colours and, if primed with suitable information about places covered by the map, can readily exploit it to act as a location-aware guide for that area.

North oriented digital maps of all parts of the UK can be obtained either from Google maps in the form of annotated satellite photos or free from the Ordnance Survey via their new OpenData service. By travelling to two separate spots on these maps that are not due north, south, east or west of each other, putting the cursor over where the user is and clicking on a button, the software in such an application can correlate the GPS sensed latitude and longitude of the user's current location with the cursor indicated X and Y screen coordinates, and so calibrate the application to correlate any GPS sensed location covered by the map with its screen presented position by simple linear correlation [2].

After that the software should be able to display the user’s position in real time on the map with an icon. A user could then interactively enter details on more significant map positions, either by going to them and keying in observed details or by getting the application to display the X and Y coordinates of significant positions and entering descriptive details on them in an editor and then uploading the data to the application’s persistent memory store. The application should use such information in various ways as a guide once it is calibrated and loaded with details on its local area. Such data could be searched for via its metadata fields to find locations or to display significant places in the user’s vicinity. The user could also be shown in what direction to travel to reach any known place.
The application should be programmable in Java, Microsoft's .NET Compact Framework or whatever programming framework is available for the device.

1.1 Aims and Objectives

The overall aim of the project is to develop an application to assist users to navigate about any location in the UK using a north-oriented digital map that exploits knowledge of where the user is, and that can provide the user with location-relevant advice and assistance. The application should support the capture of details on locations in an area as well as exploiting those details to provide location-aware guide services.

The key objectives of this project are to develop an application that is portable, configurable, location-aware, and a map guide.

1.1.1 Portable

The major importance of this application lies in its mobility as it is a mobile application, which users would just need to install and run on their GPS-enabled smartphones or handheld computers. The application can be used at any time and anywhere in the UK, as long as there is a clear view to the GPS satellites, or available Internet connectivity through wireless networks or 3G networks for triangulation of the user’s location.

1.1.2 Configurable

The user should be able to interactively enter details on more significant map positions, either by going to them and keying in observed details or by getting the application to display the X and Y coordinates of significant positions and entering descriptive details on them in an editor and then uploading the data to the application’s persistent memory store. The application could use such information in various ways as a guide once it was calibrated and loaded with details on its local area.
1.1.3 Location Awareness

The major objective of this project is that the application should be able to locate the user’s current location in real time using the inbuilt GPS in the mobile device and that the user should then be able to enter important information about any specific location. The application will work in real time to track the current location of the user. However, due to the interactive nature of the application, by travelling to a different spot on the map, either by zooming or panning and clicking on a button the application would then correlate the GPS-sensed latitude and longitude with the user’s current location. After that the application would be able to display the user’s position in real time on the map with an icon.

1.1.4 User-friendliness

The project is aimed at providing an application which is easy to use and intuitive to understand. The term ‘user-friendly’ mostly refers to an interface from which the user gets his or her desired result with ease and which does not mislead him about his destination. The user should be able to interact with the application easily. There should be no ambiguity in its interface. The interface of this application will be simple. It will take into account the following attributes to measure its user-friendliness:

- **Presentation of interesting data** – both map data and satellite data will be presented
- **No unnecessary information** – each page will display minimum relevant information
- **Complexity of data will be matched with simplicity of design** – the user will not need to know the technical details about the workings of geographic data or GPS in order to use this application.

The organisation of this dissertation is as follows:

- **Chapter 1: Introduction**: This chapter establishes the foundation of the project by describing the aim and objectives of the project. It also contains a structured overview of the dissertation document.
Chapter 2: Literature Review: In this chapter, the author describes the importance of the project and reviews the existing literature on the topic. This covers mobile computing and location sensing technologies such as the Global Positioning System (GPS) and wireless network infrastructure. Understanding the basics of the Geographic Information System (GIS) and digital mapping was invaluable in successfully undertaking this project. This chapter also presents a detailed overview of GIS and explains the nature of geographic data and the components of digital maps. This chapter further contains a study of the available mobile application development technologies that can be used to undertake the development of the application.

Chapter 3: Requirement Analysis: In this chapter, the requirements of the application are analysed. It contains both functional and non-functional requirements; the related professional, legal and ethical issues are also explained.

Chapter 4: Development Process: This chapter provides a detailed description of the development process based on the Incremental and Iterative Development approach.

Chapter 5: Testing: This chapter covers detailed testing of the application, including unit testing, integrated testing, system testing and user acceptance testing.

Chapter 6: Evaluation: This chapter evaluates the project based on feedback from the user acceptance testing.

Chapter 7: Conclusion and Recommendations: Finally, this chapter concludes the project, as well as looking to the future, providing recommendations for future work that can be implemented to further improve the application.
Chapter 2: Literature Review

In this section, the researcher will discuss the importance and significance of this project and the technologies associated with the development of the proposed application.

This project is intended to develop an application which is a configurable map guide for handheld computers or smartphones that use GPS to present location sensitive information about any chosen map vicinity in the UK.

2.1 Project Importance

This section lists some significant aspects of the project in order to prove the value and importance of this portable configurable, location-aware application. The most important features of this application are that it is:

- **Configurable**: permits registering of detailed geographic locations that the user deems significant
- **Location-aware**: senses the user’s location through the inbuilt GPS
- **Handy**: can be installed on mobile computers or smartphones and can be carried and used anywhere
- **A map guide**: acts as a geographical guide that can be built up based on the user’s activities and inputs

In what follows, the technologies associated with the development and operation of this project will be presented. These include mobile computing, geographic information systems (GIS) and digital mapping, mobile development platforms, and location sensing technologies such as the Global Positioning System (GPS) and wireless network infrastructures.
2.2 Mobile Computing

Mobile technologies such as handheld computers and smartphones have become a familiar part of our daily lives. Mobile applications are basically applications that run on smartphones or mobile devices and perform certain tasks for the user. In this modern era with the availability of sophisticated smartphones and mobile devices, mobile applications are becoming widespread because of the many extended functionalities they provide to users, ranging from simple tasks such as text messaging to more advanced tasks like navigation. Mobile applications are normally downloadable directly onto the mobile device or smartphone, mainly via Wi-Fi.

As smartphones and mobile devices become more universal and deep-rooted in our daily lives, mobile applications that combine location awareness, mobility, and mapping are providing users with many new and innovative capabilities. The development of such applications is made possible by the availability of GPS-embedded mobile devices and mostly free web-mapping services (WMS) from governments and organisations such as the Ordnance Survey and Google. These WMS provide both vector and raster geographic data in the form of satellite imagery or aerial photography. Mapping data can be huge in size. However, with the availability of WMS, this data does not need to be downloaded into mobile devices as it can be easily accessed through web services and the information can be stored as required in its memory, which makes the use of mapping data easier on mobile devices.

Various GPS tracking applications have been developed for mobile devices and smartphones, some of which are listed below.

2.2.1 GPS Tracking Apps

A company called LOCiMOBILE has developed several GPS tracking applications for smartphones; namely iLOCi2 (basic), GPS Tracking (enhanced) and Tracking (real time). These are GPS-integrated smartphone applications that allow the user both to request a location and to broadcast his/her location to other GPS Tracking users in his/her contact list [7].
2.2.2 My Tracks

My Tracks is an Android-based application which records the user’s tracks as they travel using GPS. The user can mark his or her travelling path by using the application. It also enables viewing of the user’s real-time location and the sharing of activities with friends and families via email and social networking sites like Twitter and Facebook [5].

2.3 Location Sensing Technologies

In location-aware applications, the first step is to identify the source of location information. This section will describe the main location sensing technologies that will be used to implement the location-aware features of the application for positioning real-world objects.

2.3.1 Global Positioning System (GPS)

The operation of the application of this project will be made possible by the availability of the global positioning system (GPS) integrated into handheld computers or smartphones for calculating the user’s current location.

GPS is a satellite-based system that provides users with positioning, navigation, and timing (PNT) services. GPS is owned by the United States [13] and consists of a network of 24 operational satellites (plus some spares in case of failure), each orbiting the Earth every 12 hours on distinct orbits at a height of 20,200 km and transmitting radio pulses at very precisely timed intervals [8]. GPS was originally intended for US military use. For that reason, the US government imposes Selective Availability (SA) on its operations, which can intentionally introduce random inaccuracies of up to 300 metres on civilian GPS receivers to prevent enemies or foreign troops from using the GPS navigation system to their advantage.
However, as of 2 May 2000 the US government decided to turn off the SA in the GPS system and it became widely available for civilian use with up to 10 times better accuracy than with SA [3]. The GPS system works 24 hours a day and can be accessed free of charge, anywhere in the world and in all weather conditions, provided there is a clear view between the GPS satellites and the receiver.

By analysing signals from at least four (three if elevation is not needed) of the 24 operational satellites, a GPS receiver on the Earth’s surface can compute and display the position of the receiver in three dimensions, latitude, longitude, and elevation \((x,y,z)\).

For its complete implementation, the GPS system is made up of three segments. Below is a brief overview of these three segments.
Space Segment
GPS satellites are the space segment. Radio signals are sent by the space vehicles (SVs). The GPS Operational Constellation has 24 satellites which orbit the earth every 12 hours. In fact there are more than 24 operational satellites. New satellites are periodically launched to replace the older ones.

Control Segment
The control segments consist of a system which is responsible for locating all the stations located on Earth. The control segment consists of five unmanned monitoring stations and one master control station. The five unmanned stations continuously monitor signals from the space segment and relay that information to the master control station. The master control station then processes the radio signals and sends the processed signals back to the space vehicles.

User Segment
The user segment consists of GPS receiver equipment and the user community. The GPS receiver is responsible for converting the SV signals into latitude, longitude and elevation for calculating the current location and altitude of the user. The user segment contains many thousands of commercial and recreational civilian users as well as military users around the world. In this application the GPS receiver will be a smartphone or handheld computer with an inbuilt GPS sensor that will be used to run the application.

The GPS approach is currently useful for outdoor use. However, it is ineffective indoors because the GPS radio signal is obstructed by buildings. There needs to be a clear view of the sky for the receiver to successfully communicate with the satellite system.

2.3.2 Wireless Network Infrastructure

Since GPS is limited to outdoor use only, there is a need for an alternative approach which can overcome the limitations of the GPS approach. The most appropriate alternative is the wireless network method. In this approach, the mobile computer or smartphone estimates its location based upon the locations of wireless access points,
which are fixed at known locations, or knows its location by GPS. The radio signals of the wireless communication network serve as sensors and produce information on where the mobile terminal is located. This approach is applicable for both indoor and outdoor environments and it is appealing because there are many established Wi-Fi hot-spots in the UK. Also, more importantly, to make use of this approach no further equipment is required other than access to a wireless hot-spot.

### 2.4 GIS and Digital Mapping

One key component of this project will be the integration of digital maps as the back-end of the application. Geographical Information System (GIS) is the key technology that makes it possible to integrate mapping into the application. Therefore, in this section we will give a brief introduction to GIS and some background on digital mapping.

A Geographic Information System (GIS) is a system that consists of software, hardware, and spatial and non-spatial (attribute) data for capturing, managing and analysing all forms of spatially referenced information. According to [8], a GIS is a tool for performing operations on geographic data that are too tedious, expensive or inaccurate to perform by hand. Hidden in most data is a geographical component: an address, postcode, borough, city, county, or latitude/longitude coordinate – GIS is used to facilitate the exploration of such spatial elements of data. Depending on software capabilities, users can display, query, and analyse geographic data remotely through a web browser interface or map-based application.

GIS is made up of both spatial and non-spatial data. Spatial data consists of geographic features or a phenomenon that occupies a location. Non-spatial data does not have a specific location in space; however, it can be linked to a geographic location through geocoding. For example, Heriot-Watt University is a spatial feature and the associated information about the University (name, postcode, city, country, number of students, etc.) comprises non-spatial attributes which are linked to the University by its location. GIS is made up of various themes or layers and a collection
of layers form a GIS database. For instance, in a city map all roads could make up one layer, buildings another, and land plots yet another layer.

2.4.1 Geographic Data

**Vector Data Model**
In the vector data format, locations are represented as points, lines or polygons (areas), which are encoded and stored as a collection of X and Y coordinates. The vector data format is useful for describing discrete phenomena, such as cities and rivers. The location of a point feature, such as a phone booth, can be depicted with individual X and Y coordinates. Linear features, such as rivers and roads, can be represented as a group of points via their coordinates. Area or polygon features, such as land plots, can be stored as a closed loop of coordinates.

**Raster Data Model**
The raster data format models continuous features or phenomena, such as land use or soil types. It is made up of a collection grid of cells, which are formed into an image. Satellite images and aerial photos are examples of raster data. Both vector and raster data have their advantages and disadvantages. For this project, I will use both vector and raster data and the user will be able to display whichever data type is desired. GIS data can be stored as a single feature in a file system or related features can be efficiently managed using a Relational Database Management System (RDBMS) like SQL Server or Oracle.

2.4.2 Maps

**Map Scales**
Map scale is the proportion of features on the map in relation to actual features on the Earth. Most maps are much smaller than the reality they represent, and the map scale tells us how much smaller [10]. There are basically three ways of stating map scales: as a ratio, as a written statement or as a graphical element. Ratio scales, for instance 1:50,000, relate 1 unit of measurement on the map to 50,000 units of measurement of the ground. In the ratio scale, the unit does not matter and therefore no unit is stated on the map scale. However, the units of measurement or representation must be the
same; that is, a 1 centimetre line on the map would represent a 50,000 centimetre stretch of stream on the ground. The number on the left of the ratio colon is always 1, by convention. Written scales are self-explanatory and can be written as a short statement. For examples, one inch represents one mile – this means that one inch of measurement on the map is equal to one mile on the ground. Therefore, if a stretch of road on the map measures 2 inches on the map – the actual length of the road on the ground will be 2 times 1 mile, which equals 2 miles. Graphical scales are the safest and most helpful means of representing a map scale. A simple bar scale typically portrays a series of conventional rounded distances appropriate to the map’s function and the area covered [10].

Larger scale maps represent maps that cover a lesser area in finer detail, for example, a city map at 1:5,000 scale. Small-scale maps cover a greater area in less detail – for instance, a world map at 1:1,000,000 scale. Digital maps adjust their scales as the user interacts with the map by zooming in and out to get a larger scale and smaller scale respectively. Transforming from large to small scale requires generalisation and classification of the map features: a city becomes an area, then a point; minor streets and roads are combined into a few categories; houses and the major buildings are removed; small rivers and roads are removed and less important text is also removed [6].

The scales on a printed paper map are normally fixed and relative to the size of paper that the map was printed on. For that reason, it is best to represent maps that will be printed with graphical scale bars, as the scale bar will always be in proportion to the printed map size. Therefore, accurate measurements can always be made, no matter what size of paper the map is printed on. On the other hand, the scale on an interactive map, like the one for our application, is dynamic and changes as the user zooms in/out of the map. Zooming in will produce a larger scale (more detail), whereas zooming out will produce a smaller scale (less detail).
Map Projections

Map projection is the process of transforming the curved three-dimensional surface of the Earth onto a flat two-dimensional plane. Every map projection involves a trade-off; that is, it distorts either distance, area, shape, direction, or some combination thereof. Some projections implement trade-offs, meaning that they minimize distortions in some of these properties at the expense of maximizing errors in others. Some projections attempt only to distort moderately all of these properties. Map projections fall into three main categories: planar, conical and cylindrical. To minimize distortions, these surfaces allow mapmakers to choose the most appropriate projection by centring the projection in or on the area that is covered by the map [10].
Planar projections, which are also called azimuthal projections, are used mostly for maps of the Polar Regions. Conic projections are well suited to large mid-latitude areas, such as Europe, and North America, while cylindrical projections are normally centred on the equator and are used for world maps.

**Map Symbols**

A map uses symbols to represent every feature on the map. A map symbol is a visual mark systematically linked to the data and concepts shown on a map [6]. Each symbol on a map may be represented as a point, line or polygon; the representation of a feature by any of the three map symbol types (point, line or polygon) is determined by the scale of the map. For instance, a city can be represented as a point in a small-scale map, such as a map of the world; and that same city can be represented as an area
(polygon) in a large-scale city map. Map symbols can be presented based on resemblance.

2.5 Development Technologies

Various mobile application development platforms are available for the development of this project, such as Android, Apple’s iOS, Nokia’s Symbian OS and Windows Mobile. Each of these development platforms has its own advantages and disadvantages in relation to the development of the proposed application. However, based on the advantages and disadvantages in relation to this project, the Android platform with the Java programming language will be used to develop the application. The main map data source will be Google Maps for the background spatial information. This section will explain details of the Android platform and Google Maps in relation to the aims and objectives of this project.

2.5.1 Android

Android is a suite of software for mobile devices that includes an operating system, middleware and key applications. The Android software development kit (SDK) provides the tools and application programming interfaces (APIs) necessary to develop this application on the Android platform using the Java programming language. Among its many features, the Android platform has a rich integrated development environment (IDE) which includes a device emulator, tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE [1].

Developed by Android Inc. and later acquired by Google, the Android platform allows open source development. This means that the program itself is free, and with the right ability and desire, the program can be used to suit a developer’s needs. The key to open source software is freedom—freedom to use and change a program as required [9]. While most mobile development platforms have a strict development process for applications, the Android platform allows anyone to create applications for Android-built smartphones or handheld computers.
Android is therefore the most suitable platform for developing this project. According to the official website [11] for the Open Handheld Alliance\(^1\), some of the advantages of the Android platform are detailed below:

**Open**

Android is an open platform that enables developers to create applications that utilise the inbuilt functionalities of the Android handset. It was built to be genuinely open. For instance, an application can call upon any of the device’s core functionalities, such as making calls, sending SMS, or utilising the camera. Being open source, Android can be readily extended to incorporate new cutting-edge technologies as they emerge. Also, the platform will continue to advance as the developer community works together to build innovative mobile applications [11].

**Equally created applications**

With Android, third party applications can be built to have equal access to the mobile device’s core capabilities such as making calls and sending SMS, which enhances users’ experience with third party applications and services. With Android devices, users are able to customise the device fully to suit their personal needs [11].

**Breaking down application boundaries**

With Android development, developers can combine information from the web with data on an individual’s mobile device, such as geographic location, user contacts and calendar [11].

**The Android architecture**

The Android operating system (OS) is a software stack in which each layer of the stack groups together several programs that support explicit operating system functions. The Android software stack, as shown below in Figure 2-5, is subdivided into five layers.

---

\(^1\) The Open Handheld Alliance is a consortium of 84 technology and mobile companies who have come together to develop open standards for mobile devices.
The topmost layer of the android software stack is the applications layer. This is where basic functions of the device like making calls or sending text messages can be found. This is the layer in which our application will sit for the user's operations. An Android application can basically be composed of four categories; Activity, Content Provider, Service and Broadcast Receiver. The activities of an Android application are like desktop windows (user interface) that the user uses to interact with the application. An activity has a life cycle and can be in one of five different states: Active, Pause, Stop, Restore or Destroy. An activity that is running and occupies the device screen is said to be in the active state and only one activity can be active at a time. If another activity is moved to the active state and overlaps the current active activity, then that activity will move on to a pause state and will move to the stop state once it is fully covered by the other activity. Activities in the pause and stop states can be destroyed by the system if it needs to free up memory to be used elsewhere. An activity that has been destroyed and restarted will move into the restore state and then the active state. The figure below shows the life cycle of an Android activity.
Data for certain applications are managed by a Content Provider which also controls the accessibility of the data. A Service runs in the background. Broadcast Receivers listen and respond to broadcast announcements.

The next layer in the Android architecture is the application framework. This includes the programs that manage the phone’s basic functions like keeping track of the device’s physical location using its inbuilt GPS receiver or wireless network. The Libraries layer is a set of instructions that tell the device how to handle different kinds of data. For instance, the SQLite library supports the storing of our applications data into the device memory. The Android virtual machine is software that behaves as if it is a real device with its own operating system. At the bottom of the stack is the kernel which includes Android's security settings, memory power management, and several hardware drivers.

### 2.5.2 Java Programming Language

The Java programming language is a high-level object-oriented programming language that is simple, architecture-neutral, portable, distributable, multi-threaded, robust, dynamic and secure [14]. Java is a modern software language that is currently the most widely accepted style for programs. There is an extensive collection of existing classes in library packages called Java Application Programming Interfaces.
Android applications are Java-based, which means that applications are developed using the Java programming language. Also, Android applications can be easily developed in Eclipse IDE using the Android plug-in for the Eclipse IDE. Java can be used to produce various types of applications, such as:

- Mobile computer or smartphone applications
- Desktop applications
- Applets (applications embedded in a web page)
- Server-side applications to produce dynamic web pages

### 2.5.3 Map Data Source

There are several freely available map data sources that can be used with this project. However, the two most suitable data sources are Google Maps and Ordnance Survey Open Data Map services. However, since the Android development platform is another Google product, it is obvious that Google Maps can be more suitably integrated into the Android operating system than the Ordnance Survey’s new Open Data Map service. Therefore, Google Maps will be used as the back-end map service for developing the proposed application.
All Google Maps come with their coordinates (latitude and longitude) and map scales pre-supplied. Google Maps uses the World Geodetic System 1984 datum (WGS84) and Web Mercator projection. The Google Maps service is made up of 20 zoom levels (level 0 – 19) of predefined map scales, where level 0 is the smallest scale (the least detail) and level 19 is the highest scale (the most detail). As the user zooms in and out of the application, more and less detailed map information is displayed respectively. For example, a zoom in from level 1 to level 5 will display more map information, while a zoom out from level 5 to level 1 will result in less map information. To facilitate faster rendering of the map layers, Google Maps are cached into various tiles at each scale level. This will also increase the performance of the application because the map is drawn faster from the Google Maps server.

Google Maps API Web Services
Web services for mapping essentially fill two roles: accessing remote data sources as a consumer and serving up or sharing data as a provider for others. Web services for

---

2 [http://earth-info.nga.mil/GandG/wgs84/index.html](http://earth-info.nga.mil/GandG/wgs84/index.html)
mapping are all about sharing information [8]. Google Maps is a web mapping services (WMS). This service is free for non-commercial use and developers can overlay their own data on top of it to suit their needs. Google Maps has various Application Programming Interfaces (APIs), including:

- Web Services
- Maps JavaScript API
- Maps API for flash
- Google Earth API
- Maps Image API

However, in the development of this project, I will be mainly using the Web Services API; therefore, it is useful to give some brief details of the Maps Web Services API. The Maps Web Services API is a collection of HTTP interfaces to Google services providing geographic data for maps applications. The Google Maps API provides these web services as an interface for requesting maps API data from external services and for using them within map applications [4]. There are various web services:

*Directions API*  
this is a service that uses an HTTP request to calculate the directions between locations

*Distance Matrix API*  
this is a service that provides travel distance and time for a matrix of origins and destinations

*Elevation API*  
this is a service that provides the elevation for all locations on the Earth

*Geocoding API*  
this is a service that is used to convert street addresses to actual geographic coordinates (latitude and longitude) to enable their location on a map

*Places API*  
using a HTTP request, this service returns information about known establishments, geographic locations, or prominent points of interest.

All of the above listed Maps Web Services could be used in the development of this application.
Chapter 3: Requirement Analysis

This chapter will provide detailed analyses of the requirements of the application of this project. Requirements analysis covers what the application should do and support. Successful project planning and development must include a thorough understanding of the project’s requirements. In this section, we will provide a complete analysis of the requirements of the application in this project. This project has both functional and non-functional requirements.

3.1 Functional requirements

Functional requirements are associated with specific functions, tasks or behaviours that the system must support. There are two main categories of functionalities in this project: one is the location-aware features and the other is the configurable features, as illustrated below:

![Figure 3-1: Overview of Functional Requirements](image-url)
The interface of the application will be simple and user-friendly. Users will not find any ambiguity in the design. It is important that the interface should not be complex and should be interactive. A detailed list of functionalities with different priority levels is given below. Priority levels are given according to the importance of the functions to the software. As seen from the table below, there are three levels of priorities, Low (L), Medium (M), and High (H).

Features, functionality and importance

<table>
<thead>
<tr>
<th>ID</th>
<th>Features</th>
<th>Functionalities</th>
<th>Priority</th>
<th>Implemented?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get Location</td>
<td>To display the user’s current location</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Get X and Y coordinates</td>
<td>To calculate the X and Y coordinates of any location</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Map View</td>
<td>To show a standard vector map</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Satellite View</td>
<td>To show a satellite map</td>
<td>M</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>GPS</td>
<td>To switch to GPS operation mode</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Network</td>
<td>To switch to wireless network operation mode</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Show Distance</td>
<td>To display the distance between two known points</td>
<td>M</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Show Direction</td>
<td>To show the direction to follow to get to a given destination</td>
<td>M</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>Register location</td>
<td>To register the details of a location</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Edit location details</td>
<td>To edit the details of a registered location</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Display registered locations</td>
<td>To display a list of registered locations</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Zoom In</td>
<td>To zoom in to a large map scale</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Zoom Out</td>
<td>To zoom out to a small map scale</td>
<td>H</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Share location details</td>
<td>To enable the user to share any location details with other mobile users</td>
<td>L</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 3-1: Detailed description of Functional Requirements
3.2 Non-functional requirements

Non-functional requirements are those necessities that define the qualities of the application apart from its functionalities. This project aims to achieve the following non-functional requirements:

**Usability** it is a portable mobile application. It is essential that it is easy to use with no ambiguities in its design. The interface of the application should be simple, intuitive and user-friendly.

**Stability** the application should be reliable and stable and not prone to crashing or freezing, which is common with most mobile applications that are used on smartphones because of the limited memory capacity and processing power of mobile devices.

**Extensibility** the application should be developed to be extensible so that functionalities can be readily expanded without major modification to the original code base of the application.

3.3 Professional, Legal and Ethical Issues

There are no serious legal or ethical issues associated with the development of this project; however, the following professional concerns should be taken into consideration when developing such an application:

- There is a technology risk, that another application with more advanced features and newer technology with greater compatibility might emerge
- The project should be aimed at serving the public and their needs
- The time constraint should be kept in mind while developing the project
- The project should meet the quality constraints

Users will participate in the summative evaluation of this project, but there are no special concerns regarding their privacy and wellbeing beyond what is usual in anonymised user testing of interactive software.
Chapter 4: Design and Development Process

As planned (see appendix II), the project was developed using the Interactive Incremental Development (IID) approach. Various methodologies or combinations of tools and techniques could be used. However, after a careful review of various approaches and based on the scope of this MSc project to be developed and completed by a single developer (the student) guided by a supervisor (user/stakeholder), the IID is deemed to be the most suitable approach for the development of the application. This chapter provides a brief introduction of the IID approach and its advantages in relation to the development of the project. The chapter further explains how the project was developed using the IID approach.

4.1 Methodology

As stated in Chapter 1, the overall goal of the project is to develop a mobile application that is portable, configurable and location-aware which can assist users to navigate about any location in the UK using a north-oriented digital map that exploits knowledge of where the user is and that can provide location-relevant advice and assistance about where he or she is. The application should support the capture of details on locations in an area, as well as exploiting those details to provide location-aware guide services.

As stated above, in reference to the scope of the project as a master’s project to be developed by a single developer (the student) and with limited project completion time, after a careful review of various methodologies, including the waterfall and spiral development life cycle, the Interactive Incremental Development approach was deemed to be the most appropriate for the development of this application. In Iterative Incremental Development, the requirements are broken down into increments, each providing part of the overall functionalities of the application. Each increment is developed in one iteration as a mini-application including further requirements analysis, design and development, and testing. At the end of each iteration, the application is stable, integrated, tested and partially complete. The application at the end of an iteration may be released to customers; however, this is not the case in the
development of this project. The increments have been chosen for the benefit of the developer in relation to the scope of the project.

![IID methodology diagram](image)

Figure 4-1: IID methodology – adopted from Adv. SE course note

The approach to developing this application is to develop a very simply workable application and continue to incrementally add new functionalities to enhance its usability and functionalities. According to Gall’s Law [17], “A complex system that works is invariably found to have evolved from a simple system that worked. The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a working simple system”. Unlike other methodologies such as the traditional waterfall, which only produce results at the end of the project life cycle, with the IID the application is developed, tested and deployed as a working mini-application from each iteration and can be further enhanced incrementally with new features.

In choosing the IID approach for the development of the application of this project, the following were taken into consideration:

**Scope of the project** – the scope of the project is to develop an application for a master’s project with very limited time (approximately 3 months) for completion; hence it is necessary to have a workable application as early as possible, which can at least demonstrate the view of the project and can be further developed and improved as time permits.
Production of interdependent working applications – with the IID approach a stable working application is produced for every iteration, which the developer can present to the user if required.

Single application with many functions – at the end of every iteration the requirements are reviewed and analysed and further requirements are incrementally implemented into the next application; in this case, the final single release application is enhanced and equipped with more features.

According to the initial project plan (see appendix II), using the IID approach, the project development process is divided into four phases.

- Initiation Phase
- Planning Phase
- Execution Phase
- Closing Phase

There is a specific purpose for each of these phases in successfully developing this project. This section explains these four phases and how they were achieved.

Initiation Phase
The initiation phase was the beginning of the project. This phase was to identify and explore the scope of work or project requirements while taking into account the available time and existing development skills of the developer. The project goals and objectives were agreed with the supervisor. This phase was intended to examine the feasibility of the project.

Planning phase
During this stage, the developer identifies the problem domain and analyses the requirements that are associated with the successful development of the application. Detailed use case analyses including use case textual descriptions were used.
**Execution Phase**

The principal objective of this phase is to build the actual application. During this phase the application is designed, developed and tested by following a series of iterations.

**Closing Phase**

This is the final phase of the project. In this stage the final application and any written documentation (dissertation) is submitted to the stakeholder (Heriot-Watt University).

### 4.2 Application Design

In the design of this application and to better understand the application processes, use case textual descriptions were used to provide an explanation of the processes to be performed by the application as seen from the users’ perspective [see the next sections for detail]. According to [15], a use case is a single task, performed by the end user of a system that has some useful outcome. In order to aid the development of the main graphical user interface (GUI), the requirements were further designed into a use case diagram for each increment – this is an overview of the use cases in the system, showing actors (users/stakeholders) and use cases and their interaction with the system. Based on the use case diagrams, using a GUI tool (Android GUI Stencil version 1.4), the GUI for the very frequently used and high-visibility components was mocked up.

The next sections will present the use case description for the individual tasks as seen from the users’ interaction with the application.
4.2.1 Location-aware Use Case - Requirements

This section analyses the application’s requirements using use case descriptions

Get user’s current location use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>L01</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Get user’s location</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To pinpoint the user’s current location on the map</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>High</td>
</tr>
</tbody>
</table>

Pre-conditions
The application is open

Main success scenario
1. User taps on the My Position button on the menu
2. The application correlates the user’s position via GPS or wireless network
3. The application zooms and pans to the user’s location

Exceptions
1a No GPS or network coverage
   1. User is informed of error, continues from 1.

Post-conditions
The application has opened a map of the user’s current vicinity

Table 4-1: Get user’s current location - Use Case textual description
Get X and Y coordinates use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>L02</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Get X and Y coordinates</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To ask the application to specify the X and Y coordinates of any location on the map</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>High</td>
</tr>
</tbody>
</table>

**Pre-conditions**
The application is open

**Main success scenario**
1. User taps on a location on the map
2. The application computes the X and Y coordinates of the location

**Exceptions**

**Post-conditions**
The application has displayed the X and Y coordinates of the location on the map

---

Display standard map view use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>L03</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Display Standard map</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To display standard vector map view on the application</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>High</td>
</tr>
</tbody>
</table>

**Pre-conditions**
The application is open and the main activity is active

**Main success scenario**
1. User taps on the Map button on the menu
2. The application fetches the vector map from the Google Maps server
3. The standard map is displayed

**Exceptions**
1a Mobile device does not connect to the Internet
   1. User is informed of error, continues from 1.

**Post-conditions**
The application has been displayed in standard vector map view

---

Table 4-2: Get X and Y coordinates - Use Case textual description

Table 4-3: Display map in Standard View - Use Case textual description
Display satellite map view use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>L04</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Display Satellite map</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To display satellite imagery map view on the application</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>High</td>
</tr>
</tbody>
</table>

Pre-conditions
The application is open and the main activity is active

Main success scenario
1. User taps on the Satellite button on the menu
2. The application fetches the vector map from the Google Maps server
3. The satellite imagery map is displayed

Exceptions
1a Mobile device does not connect to the Internet
   1. User is informed of error, continues from 1.

Post-conditions
The application has been displayed in satellite imagery map view

Table 4-4: Display map in Satellite View - Use Case textual description

Operate in GPS mode use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>L05</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Operate in GPS mode</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To enable the application to correlate locations using the smartphone or mobile computer’s inbuilt GPS receiver</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>High</td>
</tr>
</tbody>
</table>

Pre-conditions
The application is operating in Network mode

Main success scenario
1. User taps on the GPS button on the menu
2. The application is ready for GPS use

Exceptions
1a No GPS coverage
   1. User is informed of error, continues from 1

Post-conditions
GPS mode is enabled in the application

Table 4-5: Operate application in GPS mode - Use Case textual description
### Operate in Network mode use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>L06</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Operate in Network mode</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To enable the application to correlate locations using the smartphone or mobile computer’s 3G network or Wi-Fi</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>High</td>
</tr>
</tbody>
</table>

**Pre-conditions**
The application is operating in GPS mode

**Main success scenario**
1. User taps on the Network button on the menu
2. The application is operating in network mode

**Exceptions**
1a No 3G network or Wi-Fi coverage
   1. User is informed of error, continues from 1

**Post-conditions**
Wireless network mode is enabled in the application

### Show distance use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>L07</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Distance from A to B</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To enable the application to calculate and show the travel distance from one location to another</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**Pre-conditions**
The application is open

**Main success scenario**
1. User selects location A on the map
2. User selects location B on the map
3. The application calculates the distance from location A to location B

**Exceptions**

**Post-conditions**
The distance from location A to location B has been calculated and displayed on the map.
Show direction use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>L08</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Show Direction from A to B</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To enable the application to calculate and show the travel direction from one location to another</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**Pre-conditions**
The application is open

**Main success scenario**
1. User selects location A on the map
2. User selects location B on the map
3. The application calculates the direction from location A to location B

**Exceptions**

**Post-conditions**
The direction from location A to location B has been calculated and displayed on the map.

Table 4-8: Calculate Direction from point A to B - Use Case textual description
4.2.2 Configurable Use Case - Requirements

This section will give a detailed analysis of the configurable features using Use Cases.

**Register Location use case textual description**

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>C01</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Register location</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To enable the application to register a location on the map with detailed descriptive information on the location</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>High</td>
</tr>
</tbody>
</table>

**Pre-conditions**
The application is open

**Main success scenario**
1. User taps on a location on the map
2. User taps on the Add location button on the menu
3. User types in the details of the location and taps the Save button
4. The details are stored in the application’s persistent memory

**Exceptions**

**Post-conditions**
The location and details have been added to the list of locations in the application’s memory.

Table 4-9: Register location - Use Case textual description
### Edit location details use case textual description

| USE CASE ID: | C02       |
| USE CASE:    | Edit location details |
| GOAL:        | To enable the application to edit saved location details |
| ACTORS:      | User of the application |
| PRIORITY:    | High |

**Pre-conditions**
The application is open

**Main success scenario**
1. User taps on a location on the map
2. User taps on the Edit location button on the menu
3. User types in the new details of the location and taps the Save button
4. The details are stored in the application’s persistent memory

**Exceptions**

**Post-conditions**
The updated location and details have been added to the list of locations in the application’s memory.

---

### Display registered locations use case textual description

| USE CASE ID: | C03       |
| USE CASE:    | Display registered locations |
| GOAL:        | To enable the application to display a list of saved locations |
| ACTORS:      | User of the application |
| PRIORITY:    | High |

**Pre-conditions**
The application is open and running

**Main success scenario**
1. User taps on the List button on the menu
2. The list of saved locations is displayed

**Exceptions**

**Post-conditions**
A list of all the saved locations has been displayed to the user.

---

Table 4-10: Edit Registered location - Use Case textual description

Table 4-11: Display list of registered locations - Use Case textual description
**Zoom In use case textual description**

| USE CASE ID: | C04 |
| USE CASE: | Zoom In |
| GOAL: | To enable the application to zoom in and display more detailed geographic information on the map |
| ACTORS: | User of the application |
| PRIORITY: | High |

**Pre-conditions**
The application is open and running

**Main success scenario**
1. User taps on the Zoom In button on the menu
2. The application’s map displays more geographic detail

**Exceptions**

**Post-conditions**
The map has been zoomed in to a lesser geographic area to display more map detail.

*Table 4-12: Zoom In - Use Case textual description*

**Zoom Out use case textual description**

| USE CASE ID: | C05 |
| USE CASE: | Zoom Out |
| GOAL: | To enable the application to zoom out and display lesser geographic details on the map |
| ACTORS: | User of the application |
| PRIORITY: | High |

**Pre-conditions**
The application is open and running

**Main success scenario**
1. User taps on the Zoom Out button on the menu
2. The application’s map displays less geographic detail

**Exceptions**

**Post-conditions**
The map has been zoomed out to a wider geographic area to display less map detail.

*Table 4-13: Zoom Out - Use Case textual description*
### Share location details use case textual description

<table>
<thead>
<tr>
<th>USE CASE ID:</th>
<th>C06</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE CASE:</td>
<td>Share location</td>
</tr>
<tr>
<td>GOAL:</td>
<td>To enable the user to share any location details with other users using available media</td>
</tr>
<tr>
<td>ACTORS:</td>
<td>User of the application</td>
</tr>
<tr>
<td>PRIORITY:</td>
<td>High</td>
</tr>
</tbody>
</table>

#### Pre-conditions
The application is open and running.

#### Main success scenario
- **Chapter 1:** User taps on the Send button on the menu
- **Chapter 2:** The application displays a list of available sharing options, Email, Facebook, etc.
- **Chapter 3:** User selects appreciate sharing method
- **Chapter 4:** User sends information

#### Exceptions

#### Post-conditions
The location information has been sent to the intended users.

Table 4-14: Share location information - Use Case textual description

Android devices come with a variety of screen sizes and resolutions. Android divides the range of actual screen sizes and densities into a set of four generalised sizes, small, normal, large and extra-large. There is also a set of four densities to go with every screen size; these are low dpi (ldpi), medium dpi (mdpi), high dpi (hdpi) and extra high dpi (xhdpi). The application of this project was developed with a screen size of 3.7in Wide Video Graphic Array (WVGA) 800x480 screen resolution. The UML diagrams were developed in a physical design using the Eclipse Android IDE-integrated GUI tool. The advantage of this integrated tool over other stand-alone GUI tools is that it makes it easy to change the properties of individual tools inside the IDE using a graphical layout interface without opening the xml file.
4.3 Two Incremental Versions

The application was developed in two increments which are called Version 1.0 and Version 1.1. Using the IID approach, the requirements were broken down into increments with each increment providing part of the overall functionality. Each increment was developed in iteration as a mini-project which includes further requirements analysis, design, programming and testing. At the end of each stage, the system is stable, integrated, tested and partially completed. The increments were chosen in order of functional priority in attaining the project’s goals. Table 4-15 below shows the list of functionalities that were implemented in each version of the application.

<table>
<thead>
<tr>
<th>Increment</th>
<th>Requirements implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increment One (Version 1.0)</td>
<td>• Determine the current location</td>
</tr>
<tr>
<td></td>
<td>• Register location</td>
</tr>
<tr>
<td></td>
<td>• Display standard or satellite map view</td>
</tr>
<tr>
<td></td>
<td>• Zoom in and zoom out of map</td>
</tr>
<tr>
<td>Increment Two (Version 1.1)</td>
<td>• Version 1.0 plus</td>
</tr>
<tr>
<td></td>
<td>• Display list of registered locations</td>
</tr>
<tr>
<td></td>
<td>• Determine direction and distance to travel from location A to location B</td>
</tr>
<tr>
<td></td>
<td>• Search</td>
</tr>
<tr>
<td></td>
<td>• Send location details by SMS</td>
</tr>
</tbody>
</table>

Table 4-15: Version Increments

4.3.1 Increment One

In the initial version, version 1.0, a simple application was developed with some core functions of the application implemented. The functions were implemented based on the functional importance to the application. This section explains the requirements that are implemented in this version of the application.
This version of the application was achieved mainly by the use of the Android Location API, Google Maps and the Geo-location API. It is based on Eclipse Version: Indigo Service Release 1, Java 1.6 and Android 2.2.

To properly develop the application, the implemented requirements were presented in a use case diagram. Figure 4-2 below shows the use case diagram of the initial application (version 1.0).

![Use Case Diagram](image)

**Figure 4-2: Increment One (version 1.0) use case diagram**

These requirements were further designed in a mocked up user interface showing the main map activity of the application, as shown in Figure 4-3 below.
Figure 4-3: Increment One (version 1.0) User Interface design

**Operation Mode (high priority)**

This feature was implemented to configure the application to use either GPS or Network to determine the current location of the user. The application implements the GPS and Network location providers. Also, for a flexible selection of the best location provider, the developer used a `Criteria` object to define how the provider should be selected.

- **Net** Uses Wi-Fi or mobile network to determine the best location
- **GPS** Use the inbuilt GPS receiver on the Android device to determine the best location via satellites. This usually has better precision than a network

**Determine the Current Location of the user (high priority)**

One of the most essential functions of a location-aware application is to be able to determine the current geolocation of the user. This function can be achieved via a coarse network-based or a fine GPS-based location provider. In Android the
`android.location` package provides the API which is used to determine the current geoposition of the user.

**Register Location (high priority)**

By clicking on any location of interest on the map, the user is able to register the location information in the application’s persistent memory store. Once it is registered, the user can also select a register location to edit the details of the location.

**Displaying Standard (high priority)**

In order to enable the application to run faster even with slower Internet connectivity, by default the application implements the standard map view. However, the user is allowed to configure the application to use either the standard map view or the satellite view as required.

**Displaying Satellite Map View (medium priority)**

The application displays the standard vector map view by default; however, the user is allowed to enable the application to use the satellite imagery map view as may be deemed necessary.

**Zoom-in and Zoom-out of map (high priority)**

This was accomplished by activating the Google Maps inbuilt zoom controls. The `MapView` class allows the inbuilt Zoom controls to be programmatically activated through the `setBuiltInZoomControls` method call.

This version of the application was tested using the Android emulator and deployed as a mini-project within the overall requirements of the project. To accomplish this, the application was tested in Eclipse with the Android emulator using telnet to mock up a GPS location.
The initial application was tested to ensure that all the implemented requirements were functioning as desired and to detect any initial error that required fixing. Figures 4-4 and Figure 4-5 above show the standard and satellite map views respectively. The red dot at the centre of the map is the current location of the user which was mocked up by a telnet geo fix command using the Windows command prompt.

The application was also tested to register the location and its details. Figure 4-6 below shows the added location which is shown by a graphical blue flag on the map (note the appearance of the Add Fav button on the menu) which leads to the details activity that enables the user to add both a name and other information about the location that has been added to the map (Figure 4-7).
Figure 4-6: Graphic view of added location

Figure 4-7: Add details activity

Figure 4-8 below is a display of the added location with its name. All added details are editable; the editing functionality was also successfully tested.

Figure 4-8: Added location displayed on the map
In summary, the initial application was very simple. It could operate the application in GPS or network mode, locate the user’s current location, add locations to the map, edit registered locations, display the map in both standard and satellite imagery views and zoom in and out of the map.

4.3.2 Increment Two

After the successful implementation and testing of the initial application in increment one, the requirements of the application were further analysed, designed and developed to include additional functionalities. This phase of the development is called Version 1.1. As shown in Figure 4-9 below, the initial application (version 1.0) was further developed into version 1.1 by performing further requirements analysis, design, development and testing.
As shown in Figure 4-10, the incremental requirements were further designed in a mocked up user interface to get a better understanding of the implementation of the new functionalities.
The following requirements were incrementally developed into the initial application:

*Favourites List* *(high priority)*
This feature of the application enables the user to display a list of all the locations that have been saved into the applications database. From the display list the user is able to perform further actions like editing the details of a location or sending the location details by any available means.

*Direction and Distance* *(medium priority)*
This is achieved by using the Google Maps Directions and Distance Matrix APIs. The Directions API is a service that uses an HTTP request to calculate the directions between locations, and the Distance Matrix API is a service that provides travel distance and time for a matrix of origins and destinations. By clicking on the origin (Location A) on the map and then clicking on the destination (Location B), the application will calculate and display the direction to travel from Location A to Location B and calculate and display the distance (in kilometres) between the two locations and the estimated time that is required to travel the route.
**Search or Find (high priority)**

This feature enables the user to do a global search within the application. The Geocoding API and Places API were implemented to achieve this requirement. The Geocoding API is used to convert street addresses to actual geographic coordinates (latitude and longitude) to enable their location on the map, while the Places API uses an HTTP request to return information about known establishments, geographic locations, or points of interest, including locations that have been stored in the application’s memory.

**Sharing location details (low priority)**

By selecting a location on the map, the user is able to send that information by using any of the mobile device’s messaging media, such as Email, SMS, Skype, and even social networking sites such as Facebook and Twitter. Furthermore, the application will allow the user to add any attachment onto the message that is possible with the chosen sending application. For instance, sending by messaging would allow the user to attach pictures stored in the device’s memory or to use the device’s camera to take a new picture to be attached.
Chapter 5: Testing

Testing is an integral part of any software development process. In fact, with the Iteration and Incremental Development approach, testing is performed at the end of every development phase to ensure that the mini-project is complete and functional as required for that stage of the development. This part of the process ensures that any defect in the application is detected as soon possible before proceeding to the next incremental development phase. After all the development was completed, the final application was tested by the developer using both the Android emulator and a real Android mobile device. To fully test the application, the following tests were performed on the application:

5.1 Unit Testing

Every functionality of the application was tested separately in Eclipse using the JUnit testing framework. JUnit is the testing framework for Java applications, which is integrated in the Android development environment. The goal of the unit testing is to isolate each program part to ensure that it is working properly as required (please see appendix V).

5.2 Integrated Testing

The developer performed an integrated test of the entire application using the Android emulator. In order to achieve this, various Android virtual devices (AVD) with different hardware configurations were created and the application was tested within Eclipse. The application was successfully tested with the following virtual device configurations:

- HVGA 160 dpi
- WVGA 160 dpi
- WVGA 240 dpi
- WQVGA 120 dpi
In order to simulate the location-aware functionalities of the application, the developer uses a telnet client to mock up the GPS capabilities of the application. This was accomplished with the following telnet commands; the mock-up location used is London, United Kingdom.

```
telnet localhost 5554
geo fix 0.128 51.507
```

Figures 5-1 and 5-2 below show the application displayed in both the standard map view and the satellite view. The red dot at the centre of the map indicates the user’s current location as derived from the above stated mocked up location (0.128, 51.507).

Further screenshots from the complete integrated testing can be found in appendix I.
5.3 System Testing

To ensure that the application functions as required and is ready for the user acceptance test, the developer deployed and thoroughly tested the application on a real Android device. The device used was the Samsung Galaxy tab with a 7.0 inches WSVGA screen with inbuilt A-GPS for positioning and running the Android 2.2, Froyo operating systems.

In order to achieve the desired result of the testing, which is oriented to detect any defect in the application as early as possible before the application is released to the users, it is important to perform the test based on operation of the system in controlled conditions following a structured test plan. The developer conducted the test to validate the application against acceptable criteria which are displayed in Table 5-1 below.

Figure 5-3: Application running on Samsung Galaxy Tab
<table>
<thead>
<tr>
<th>ID</th>
<th>Function</th>
<th>Actions to perform</th>
<th>Desired Result</th>
<th>Result/Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Location-Aware Features</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Get Location</td>
<td>- Tap on the My Position button</td>
<td>The application pans to the user’s current location</td>
<td>Pass/H</td>
</tr>
<tr>
<td>2</td>
<td>Get X and Y coordinates</td>
<td>- Tap on a location within the map</td>
<td>The X and Y coordinates of the location are displayed on the map</td>
<td>Pass/H</td>
</tr>
<tr>
<td>3</td>
<td>Map View</td>
<td>- Tap on the Map button</td>
<td>The map changes to standard vector map view</td>
<td>Pass/H</td>
</tr>
<tr>
<td>4</td>
<td>Satellite View</td>
<td>- Tap on the Sat button</td>
<td>The map display changes to satellite map view</td>
<td>Pass/H</td>
</tr>
<tr>
<td>5</td>
<td>GPS</td>
<td>- Tap on the GPS button</td>
<td>The application is enabled to use GPS (this is the default)</td>
<td>Pass/H</td>
</tr>
<tr>
<td>6</td>
<td>Network</td>
<td>- Tap on the Net button</td>
<td>The application is enabled to use Network</td>
<td>Pass/H</td>
</tr>
<tr>
<td>7</td>
<td>Show Distance</td>
<td>- Tap on the Direction button - Tap on Location A - Tap on Location B</td>
<td>The distance from points A to B (in km) is displayed on the map</td>
<td>Pass/M</td>
</tr>
<tr>
<td>8</td>
<td>Show Direction</td>
<td>- Tap on the Direction button - Tap on Location A - Tap on Location B</td>
<td>The highlighted direction from point A to B is displayed on the map</td>
<td>Pass/M</td>
</tr>
<tr>
<td></td>
<td><strong>Configurable Features</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Register location</td>
<td>- Tap on the location on the map - Tap on the Add button - Type in the details for the location and tap Register</td>
<td>Location is registered in the application’s persistent memory with a graphic flag displayed on the map</td>
<td>Pass/H</td>
</tr>
<tr>
<td>10</td>
<td>Edit location details</td>
<td>- Select the location (flag) on the map to be edited - Tap on the Edit button - Type in the details for the location and tap Register</td>
<td>The edited details are saved to memory</td>
<td>Pass/H</td>
</tr>
<tr>
<td>11</td>
<td>Display registered locations</td>
<td>- Tap on the Add List button</td>
<td>A list of all registered locations is displayed</td>
<td>Pass/H</td>
</tr>
<tr>
<td>12</td>
<td>Zoom In</td>
<td>- Tap on the Zoom-In (+) on the zoom control on the map</td>
<td>The map zooms in to a larger (detailed) scale</td>
<td>Pass/H</td>
</tr>
<tr>
<td>13</td>
<td>Zoom Out</td>
<td>- Tap on the Zoom-Out (-) on the zoom control on the map</td>
<td>The map zooms out to a smaller (less detailed) scale</td>
<td>Pass/H</td>
</tr>
<tr>
<td>14</td>
<td>SMS location details</td>
<td>- Select the location (flag) on the map to be Sent - Tap on the Send Location button - Enter the phone number to send the information to - If necessary add any required information including attachments - Tap on the Send button</td>
<td>The information is sent to the intended mobile number</td>
<td>Pass/L</td>
</tr>
</tbody>
</table>

Table 5-1: Structured Test Plan

51
5.4 User Acceptance Testing

After the developer had completed a detailed system testing of the application on a real device, it was time to perform the user testing to see if the users were satisfied with the application. After little or no training, ten users were selected to test the application which was deployed on a Samsung Galaxy Tab (7.0”). This testing was also intended to test some of the non-functional requirements of the project. In order to accomplish this, the users were asked to perform a number of key tasks that exercise key capabilities of the application, while their performance was monitored. Upon completion, they were asked to fill in a questionnaire asking a series of questions about their experiences with the application. These questions addressed both the functionality and usability of the application and asked both open and closed questions. The users’ responses formed the basis of the evaluation process of the application.
Chapter 6: Evaluation

6.1 Evaluation Approach

The responses from the user acceptance testing formed the basis of the summative evaluation of the application. The developer approached a random sample of users with varying skills in using mobile applications and asked for their participation in testing and appraising the application. These samples were selected because they are the target users of the application. In order to obtain a proper evaluation mechanism for the application, the developer designed a two-fold approach. Firstly, after a brief explanation of the purpose and functions of the application by the developer, the users were given a task sheet containing a list of tasks for them to perform with the application. These tasks were chosen in order to exercise key capabilities of the application. The second part was the evaluation, which asked both open and closed questions about the tasks performed by the users. The evaluation was to address both the functionality and non-functional capabilities of the application. This chapter presents a summative evaluation of the results of the participants’ assessment and discusses and responds to notable issues that arise.

6.2 Participants

As stated above, the sample of the evaluation was selected based on well-defined selection criteria that include varying levels of expertise with mobile applications, age range and gender equality. The rationale for the diverse sample was to collect feedback and technical evaluation of the application from the intended users, representing a real world situation where anyone is expected to be able to use the application with little or no training. Due to the researcher’s need for as diverse a sample as possible; the participants were distributed roughly evenly, with 60% MALE and 40% FEMALE.
As shown in Figure 6-2 below, there were also varying age ranges in the sample. The majority of the participants, 40%, were aged 25-34 YEARS. 10% of participants were below the age of 18 YEARS, while 20% were 18-24 YEARS. 10% of the sample were 45 YEARS OR ABOVE. This clearly reflects the researcher’s desire to have as diverse an age sample as possible.

In relation to the participants’ skills level with mobile applications, the statistics illustrated that the majority, 60% of the sample, were AVERAGE users of mobile
applications while 20% were ADVANCED users. 10% were either BEGINNER or EXPERT users of mobile applications respectively.

![What is your skill level with mobile applications?](image_url)

Figure 6-3: Participants’ skill levels [Evaluation analyses]

This skills diversity was purposely targeted by the researcher, as this was necessary in order to evaluate the application against a variety of skills level to ensure that the application is tested by the intended users as in a real world situation.

### 6.3 Questionnaire

Each of the participants testing the application was asked to fill in a form (see appendix III) at the end of the experiment. The questionnaire was designed by the researcher to ask both open and closed questions in order to obtain complete feedback on the application. The questionnaire has three parts; the first part was to collect general information about the participants, such as gender, age and skill level with mobile applications, and to confirm whether they had successfully completed all the tasks given on the task sheet; this was in order to determine and highlight any difficulties that they faced. The second part of the questionnaire presented six closed questions with optional responses in a likert scale. This area of the questionnaire was to collect the participants’ feedback on the application’s usability and functionality. The final part of the questionnaire asked two open-ended questions in order to allow
the participants to include more information, including suggestions and their feelings about the application.

The questionnaire was an invaluable part of evaluating the application, as every participant filled in the questionnaire and handed it back to the developer. The questionnaire evaluated both functional and non-functional requirements of the application. This section will provide a comparative assessment of all the responses from the sample.

6.4 Evaluation of Results

In response to whether the participant feels that he/she had successfully completed all the tasks on the tasks sheet, the participants were given a choice of answering YES or NO; a large majority, 80%, responded YES while 20% answered NO. The question further asked the participants who answered NO to the first part of the question to list the tasks that they could not complete successfully. The responses to that part of the question showed that the difficulty was with tasks 9 and 10 on the task sheet (see appendix III).

![Pie chart showing the responses to the task completion question]

Figure 6-4: Tasks completion [Evaluation analyses]
However, the researcher further evaluated that the 20% of the participants who had some form of difficulty with the application were directly correlated with the sample who were beginners or averagely skilled with mobile applications. Since a large majority of the participants, 80%, successfully completed all the tasks on the task sheet, this indicated that the exercise was successful.

The second part of the questionnaire contained six statements that were presented with a likert scale containing five levels of possible responses: STRONGLY DISAGREE, DISAGREE, NEUTRAL (neither agree nor disagree), AGREE and STRONGLY AGREE. This part of the questionnaire was to allow the user to evaluate the application’s usability.

Figure 6-5 shows that 60% of the participants AGREE that the application is helpful while 40% STRONGLY AGREE that the application is helpful.

![The application is helpful](image)

**Figure 6-5: The application’s helpfulness [Evaluation analyses]**

It is clear from the above statistics that the application is helpful, that the users found it useful, and that they were willing and ready to use it.

As shown in Figure 6-6 below, in response to the statement that the application is easy to use, 80% of the participants AGREE while 20% STRONGLY AGREE. Therefore,
based on the questionnaire responses, the application is generally considered easy to understand and use.

In response to the statement that the application has a user-friendly interface, 70% of the participants AGREE. Furthermore, 30% STRONGLY AGREE with the statement. This further emphasizes that the application is simple, clear and easy to understand.
As illustrated below in Figure 6-8, when asked to evaluate whether the menu items were well organised and functions were easy to find, 60% of the participants AGREE, while 20% STRONGLY AGREE. However, 10% were NEUTRAL (neither agree nor disagree) while 10% DISAGREE. Nevertheless, upon further investigation it was also observed that the 10% who disagreed were directly correlated with the 10% of beginners. Based on the overall statistics, it is safe to say that the application fulfills its simplicity and usability requirement.

![Chart showing the distribution of responses](chart.png)

**Figure 6-8: Organisation of menu items [Evaluation analyses]**

As mobile devices come with limited memory and processing powers, it was essential to develop an application that will run properly on low powered devices with no problems. In this regard, 50% of the participants AGREE that the application runs smoothly with no errors while 50% STRONGLY AGREE with the assessment.
The final part of the questionnaire presented open-ended questions intended to encourage the participants to respond with more information, including their feelings about and recommendations for the application. The open-ended section contains the following questions:

- What is your overall opinion of the application?
- What recommendations do you have for improving the application?

Below is a list of some notable responses to the above questions:

**Opinions on the application**

- This is an excellent idea; while very helpful, it is also fun to use
- The application is simple to use and I like the idea that the user is in control
- The application is very light-weight and runs faster

**Recommendations for improvement**

- “I would like to take pictures of the location in addition to recording the geographic and attribute information”
- “This is a very good idea; however, it would be nice if it was possible to type in a known address and let the application provide the directions”
• “It would be nice to have a help page so that users can quickly learn how to use the application”
Chapter 7: Conclusion and Recommendations

7.1 Conclusion

The main aim of this project was to develop a mobile application to provide its users with the ease of locating places and their current location in real time. The application, which was developed for Android devices, is a configurable, location-aware application that will serve as a portable map guide for users to make sound decisions based on geographic awareness. As well as guiding users through navigating places, the application also shows the directions which the user needs to follow in order to reach a destination.

It was challenging to complete the project because of the limited time available and the many new technologies that had to be mastered and used to develop the application. However, these challenges helped to produce the application that was required to achieve the goal of the project. In order to attain the goal and objectives of the project, the developer adopted a chronological approach (as described in Chapter 1). In conclusion, although there are recommendations for future work that would improve the application (see the next section), the developer believes that the goal of developing a portable map guide for users in the United Kingdom has been successfully achieved. Furthermore, the developer now has greater knowledge of project management and has developed and improved several new skills, such as time management, by using tools like activity logs and treasure mapping.

7.2 Recommendations for Further Work

As can be seen from the users’ responses to the acceptance testing, there are many other potential functionalities that could improve the application. However, due to the limited time available for the completion of this project, the developer could not implement all of these recommended features. The list below describes features that might possibly be added to this application:
Taking pictures of an added location
As recommended by some of the participants, an integrated camera feature would greatly enhance the application. This would enable users to take photos in association with location details. This is particularly helpful as the users would be able to visually associate the coordinates of locations with the places that they have visited. However, with the current release of the application (version 1.1), it is possible to take pictures and send them as attachments while sharing location detail.

Inputting known location address to navigate
The application would be greatly improved by the addition of a navigation function that would enable the user to input the address of a known location to navigate to. This would enable the user to specify a location either by its geographic coordinates in different formats (lat/lon, UTM or MGRS) or by its physical address, and the application would be able to calculate the route based on the user’s current location. This would be particularly useful if the user knows the coordinate or address that he/she wants to go to but does not know the physical location and cannot identify it on the map.

Help/How-To
As recommended by some of the participants, it seems necessary to have a help or how-to page to explain the basic operations of the application and give instructions on how to perform the different tasks.

Recording tracks
Further enhancement could be to enable the application to be able to record tracks of the user’s trips. As the user moves about the application should be able to record the user’s tracks and store it in its memory. The user can later select a previously saved track to follow, which would be displayed on the map indicating the start and end points of the track. The user can then follow the direction of the track to go to the point of origin or the destination. For privacy reasons, the feature should be turn on or off by the user.
References


Appendix I: Application Screenshots

Adding a location

Graphic and list view of registered locations
Editing a registered location

Graphic and list view of edited locations
**Sending location details by SMS**

As shown above, by clicking on a location on the map or selecting a registered location, the user is able to send by SMS the location information which will capture the actual address of the location that is stored in the Google Maps server. Also, the user can edit the information by adding whatever details are required.
Calculating distance and direction from location A to B

Searching (On samsung galaxy tab)
Shearing location information (On samsung galaxy tab)
Appendix II: Initial Project Plan

In order to achieve the overall aims of this project, the project is divided into various parts to plan the project development, and each task is divided into further sub-tasks. Each part aims to produce a workable mini-project which fulfills the incremental requirements. The project plan for this application is:

Initiation Phase

- Select a project
- Identify project’s requirements
- Identify system boundary - considering the available time and existing development skills
- Deadline: 9th December 2011 - Project allocation
- Start date: 18th November 2011
- End date: 9th December 2011
- Project allocation: agree on project with a supervisor

Planning Phase

- Analyse requirements – identify use cases
- Prioritise requirements
- Create design
- Write Research Report
- Deadline: 2nd April 2012 - Research Report
- Start date: 1st February 2012
- End date: 30th March 2012
- Deliverables: Research Report - electronic copy through Vision and hard copy to supervisor – milestone
Execution Phase

- Develop application
- Test application on device
- Further developments and testing – increments
- Evaluate software
- Write dissertation – milestone
- Start date: 7\textsuperscript{th} May 2012
- End date: 10\textsuperscript{th} August 2012
- Deliverables: a working configurable, location-aware UK maps guide application

Closing Phase

- Submit application to Heriot-Watt University
- Submit dissertation
- Prepare poster presentation
- Deadline: 18\textsuperscript{th} August 2012 - MSc dissertation and Poster Session
- Start date: 10\textsuperscript{th} August 2012
- End date: 10\textsuperscript{th} August 2012
- Deliverables: MSc dissertation and Poster Session
Appendix III: Evaluation Tasks Sheet and Questionnaire

UK Map Guide Application Evaluation

Tasks Sheet

The developer would like to thank you for generously volunteering your time to participate in this user acceptance testing. Your responses will provide invaluable information to help me evaluate my MSc project.

Please perform the following tasks with the UK Map Guide application and respond to the attached questionnaire overleaf.

1. Run the UK Map Guide Application
2. Switch the map display to Satellite view
3. Switch the map to display Standard Map view
4. Operate the application in Network mode
5. Operate the application in GPS mode
6. Add a location (call it loc1) on the map and enter any details you wish
7. Add another location (call it loc2) on the map well away from loc1
8. Zoom in and out of the added locations
9. Calculate the direction from loc1 to loc2
10. Edit loc1 and call it loc3
11. Display your current location on the map
12. Display the list of added locations
13. Send loc2 details by SMS to your mobile phone number
14. Exit the application
UK Map Guide Application Evaluation

Questionnaire

The developer would like to thank you for generously volunteering your time to participate in this user acceptance testing. Your responses will provide invaluable information to help me evaluate my MSc project.

What is your Gender?  □ Male  □ Female

What is your Age Group?  □ Below 18  □ 18-24  □ 25-34  □ 35-44  □ 45+

What is your level of skills with mobile apps? □ Beginner □ Average □ Advanced □ Expert

Do you feel that you successfully completed all the tasks on the task sheet? □ Yes □ No

If No please list the task numbers of the tasks that you could not complete:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The application is helpful</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>The application is easy to use</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>The application has a user-friendly interface</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>The menu items were well organized and functions were easy to find</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>I immediately understood the function of each button</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>The application runs smoothly with no errors</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

What is your overall opinion of the application? .................................................................................................................................

What recommendations do you have for improving the application? ..........................................................................................................

..............................................................................................................................................................................................................................................
Appendix IV: Gantt Chart

![Gantt Chart Image]
Appendix V: Codes (Main Activity)

```java
package uk.hw.ac.djk6;

import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Timer;
import java.util.TimerTask;
import uk.hw.ac.djk6.R;

import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.PointF;
import android.location.Address;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.Handler;
import android.telephony.SmsManager;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.Toast;

import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.MapActivity;
import com.google.android.gms.maps.MapController;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.Overlay;

@SuppressWarnings("unused")
public class UNMapGuideActivity extends MapActivity {

    public static final String PRECIP_NAME = "GPDMapPreferencesFile";

    private MapController mapController;
    private MapView mapView;
```
private LocationManager locationManager;
private String provider;
private Button m_button_mode, m_button_myloc, m_button_regloc, m_button_provider,
        m_button_search, m_button_search_next;
private Button m_button_direction, m_button_sendloc, m_button_favlist;
private View m_textname;
private int getdirectionflag = 0;
private List<GeoPoint> directionpoint = new ArrayList<GeoPoint>();
private Road mRoad = null;
private int mode = 0;

private GeoUpdateHandler locationUpdater;
private GeoUpdateHandleMove locationUpdaterNOMove;

private Timer mTimer;
private GeoPoint currentSelect = null;
private UMKMapGuideDatabaseAdapter dbHelper = null;
private List<GeoPoint> searchResult = new ArrayList<GeoPoint>();
private List<String> searchResultName = new ArrayList<String>();
private int searchResultSelectIndex = -1;

public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    dbHelper = new UMKMapGuideDatabaseAdapter(this);
    dbHelper.open();
    locationUpdater = new GeoUpdateHandler();
    locationUpdaterNOMove = new GeoUpdateHandleMoveHandler();
    setContentView(R.layout.main); // bind the layout to the activity
    m_button_mode = (Button) this.findViewById(R.id.button_mode);
    m_button_mode.setOnClickListener(new Button.OnClickListener() {
        //@Override
        public void onClick(View v) {
            mode ++;
            if (mode > 5) mode = 5;
            switch (mode) {
            case 0:
                mapView.setSatellite(false);
                mapView.setStreetView(false);
                Toast.makeText(getBaseContext(),
                        "Map View",
                        Toast.LENGTH_SHORT).show();
                m_button_mode.setText("Sat");
                break;
            case 1:
                mapView.setSatellite(true);
                mapView.setStreetView(false);
                Toast.makeText(getBaseContext(),
                        "Satellite View",
                        Toast.LENGTH_SHORT).show();
                m_button_mode.setText("Map");
                break;
            case 2:
                Toast.makeText(getBaseContext(),
                        "Hybrid View",
Toast.LENGTH_SHORT).show();
    mapView.setSatellite(true);
    mapView.setStreetView(true);
    m_button_mode.setText("Hybrid");
    break;
    default:
        break;
    }
}

m_button_myloc = (Button) this.findViewById(R.id.button_myloc);
m_button_myloc.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        location location = locationManager.getLastKnownLocation(provider);
        if (location != null) {
            int lat = (int) (location.getLatitude() * 1000);
            int lng = (int) (location.getLongitude() * 1000);
            GeoPoint point = new GeoPoint(lat, lng);
            mapController.animateTo(point);
            mapController.setCenter(point);

            // add marker
            MapOverlay mapOverlay = new MapOverlay();
            mapOverlay.setPointToDraw(point, location.getAccuracy());
            List<Overlay> listOverlays = mapView.getOverlays();
            listOverlays.clear();
            listOverlays.add(mapOverlay);

            mapView.invalidate();
        } else
            locationManager.requestLocationUpdates(provider, 0,
                0, locationUpdater);
    }

    private GeoPoint point(String string) {
        // TODO Auto-generated method stub
        return null;
    }
},

m_button_reloc = (Button) this.findViewById(R.id.button_reloc);
m_button_reloc.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
        editor edit = settings.edit();
        editor.putBoolean("selectPoint", (currentSelect != null));
        if (m_button_reloc.getText().equals("Edit")) {
            editor.putBoolean("editPoint", true);
        } else {
            editor.putBoolean("editPoint", false);
        }

        editor.putInt("selectPointLat", currentSelect.getLatitudeS6());
        editor.putInt("selectPointLong", currentSelect.getLongitudeS6());

        // other code continues
    }
};
else {
    editor.putInt("selectPointLat", 0);
    editor.putInt("selectPointLong", 0);
}
editor.commit();
Intent i = new Intent(v.getContext(), LocationActivity.class);
startActivity(i);
_m_button_regloc.setText("Add");
currentSelect = null;
});

m_button_favlist = (Button) this.findViewById(R.id.favList);
_m_button_favlist.setOnClickListener(new Button.OnClickListener() {
    // @Override
    public void onClick(View v) {
        Intent i = new Intent(v.getContext(), LocationListActivity.class);
        startActivity(i);
    }
});

m_button_provider = (Button) this.findViewById(R.id.button_provider);
_m_button_provider.setOnClickListener(new Button.OnClickListener() {
    // @Override
    public void onClick(View v) {
        if (provider.equals(LocationManager.GPS_PROVIDER)) {
            Criteria criteria = new Criteria();
            provider = locationManager.getBestProvider(criteria, true);
        } else {
            _m_button_provider.setTextColor(Color.RED);
        }
        provider = LocationManager.GPS_PROVIDER;
        _m_button_provider.setTextColor(Color.GRAY);
    } else {
        provider = LocationManager.GPS_PROVIDER;
        _m_button_provider.setTextColor(Color.RED);
    }
});

m_button_search = (Button) this.findViewById(R.id.button_search);
_m_button_search.setOnClickListener(new Button.OnClickListener() {
    // @Override
    public void onClick(View v) {
        Intent intent = new Intent(v.getContext(), SearchActivity.class);
        startActivityForResult(intent, 1);
    }
});

m_button_search_next = (Button) this.findViewById(R.id.button_search_next);
_m_button_search_next.setEnabled(false);
_m_button_search_next.setOnClickListener(new Button.OnClickListener() {
    // @Override
    public void onClick(View v) {
        if (searchResult != null && searchResult.size() > 0) {
            if (searchResultSelectIndex < 0)
                searchResultSelectIndex = 0;
            else if (searchResultSelectIndex >= searchResult.size() - 1)
                searchResultSelectIndex = 0;
            else

searchResultSelectIndex ++;

geopoint p = searchResult.get(searchResultSelectIndex);
mapController.animateTo(p);
mapController.setCenter(p);
}
};

m_button_direction = (Button) this.findViewById(R.id.direction);
m_button_direction.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        getDIRECTIONflag = 1;
        DIRECTIONpoint.clear();
        m_button_direction.setText("Loc A");
    }
});

m_button_sendLoc = (Button) this.findViewById(R.id.sendLoc);
m_button_sendLoc.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (currentSelect != null) {
            return;
        }
        Geocoder geoCoder = new Geocoder(getActivity(),
                Locale.getDefault());
        try {
            List<Address> addresses = geoCoder.getFromLocation(
                    currentSelect.getLatitudeE6() / 1E6, currentSelect.getLongitudeE6()
                    / 1E6,
                    1);
            String add = "",
            if (addresses.size() > 0) {
                for (int i = 0; i < addresses.size(); i++)
                    add += addresses.get(i).getAddressLine(i) + "\n",
            }
            final Intent emailIntent = new Intent(Intent.ACTION_SEND);
            emailIntent.setType("text/plain");
            emailIntent.putExtra(Intent.EXTRA_TEXT, add);
            m_button_sendLoc.getContext().startActivity(emailIntent, "");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
};

// create a map view
RelativeLayout linearLayout = (RelativeLayout) findViewById(R.id.mainlayout);
mapView = (MapView) findViewById(R.id.mapview);
mapView.setBuiltInZoomControls(true);//set the zoom controls
mapView.getZoomButtonsController().setAutoDismissed(false);
mapView.setSatellite(false);
mapView.setStreetView(false);
mapController = mapView.getController();
mapController.setZoom(14); // Zoom 1 is world view
locationManager = (locationManager) getSystemService(Context.LOCATION_SERVICE);
Criteria criteria = new Criteria();
provider = locationManager.getLastKnownLocation(provider);
if (provider.equals(LocationManager.GPS_PROVIDER))
    m_button_provider.setTextColor(Color.RED);
else
    m_button_provider.setTextColor(Color.BLACK);
Location location = locationManager.getLastKnownLocation(provider);
if (location != null) {
    int lat = (int) location.getLatitude() * 1E6;
    int lng = (int) location.getLongitude() * 1E6;
    GeoPoint point = new GeoPoint(lat, lng);
    mapController.animateTo(point);
    mapController.setCenter(point);

    // add marker
    MapOverlay mapoverlay = new MapOverlay();
    mapOverlay.setPointToDraw(point, location.getAccuracy());
    List<Overlay> listofoverlays = mapView.getOverlays();
    listofoverlays.clear();
    listofoverlays.add(mapoverlay);
    mapView.invalidate();
}
locationManager.requestLocationUpdates(provider, 0,
0, locationUpdater);

private void getLocationOnMap() {
    // TODO Auto-generated method stub
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == RESULT_OK || requestCode == 1) {
        String msg = data.getStringExtra("searchText");
        Geocoder geoCoder = new Geocoder(getBaseContext(),
Locale.getDefault());
        try {
            List<Address> addresses = geoCoder.getFromLocationName(msg, 10);
            searchresult.clear();
            searchResultName.clear();
            if (addresses != null && addresses.size() > 0) {
                Toast.makeText(getBaseContext(), addresses.size() + "", Toast.
LENGTH_SHORT).show();
                for (int i = 0; i < addresses.size(); i++) {
                    GeoPoint p = new GeoPoint( (int) (addresses.get(i).getLatitude() *
1E6),
(int) (addresses.get(i).getLongitude() * 1E6));
                    searchResult.add(p);
                    searchResultName.add(addresses.get(i).getAddressLine(0));
                if (i == 0)
                    
                    mapController.animateTo(p);
mapController.setCenter(p);
}

searchResultSelectIndex = 0;
_@button_search_next.setEnabled(true);
}else {
_@button_search_next.setEnabled(false);
searchResult.clear();
searchResultName.clear();
Toast.makeText(getBaseContext(), "Location not found", Toast.LENGTH_SHORT)
        .show();
}
}
catch (IOException e) {
    e.printStackTrace();
}

}

protected void onResume() {
    super.onResume();
    locationManager.requestLocationUpdates(provider, 0, 0, locationUpdater);
}

@Override
protected boolean isRouteDisplayed() {
    return false;
}

class MapOverlay extends Overlay {
    private GeoPoint pointToDraw;
    private float acc = 0;

    ArrayList<GeoPoint> mPoints = null;

    public MapOverlay() {
        if (mRoad == null) return;
        if (mRoad.mRoute.length > 0) {
            mPoints = new ArrayList<GeoPoint>();
            for (int i = 0; i < mRoad.mRoute.length; i++) {
                mPoints.add(new GeoPoint(
                    (int) (mRoad.mRoute[i][1] * 1000000),
                    (int) (mRoad.mRoute[i][0] * 1000000)));
            }
            int moveToLat = (mPoints.get(0).getLatitudeE6()) + (mPoints.get(mPoints.size() - 1).getLatitudeE6() - mPoints.get(0).getLatitudeE6()) / 2;
            int moveToLong = (mPoints.get(0).getLongitudeE6()) + (mPoints.get(mPoints.size() - 1).getLongitudeE6() - mPoints.get(0).getLongitudeE6()) / 2;
            GEOPoint moveTo = new GEOPoint(moveToLat, moveToLong);
        }
    }

    public void setPointToDraw(GeoPoint point, float accuracy) {
        pointToDraw = point;
        acc = accuracy;
    }
public GeoPoint getPointToDraw() {
    return pointToDraw;
}

public void drawPath(MapView mv, Canvas canvas) {
    if (mPoints == null || mPoints.size() == 0)
        return;
    int x1 = -1, y1 = -1, x2 = -1, y2 = -1;
    Paint paintText = new Paint();
    paintText.setStrokeWidth(1);
    paintText.setARGB(255, 255, 255, 255);
    paintText.setStyle(Paint.Style.FILL);
    paintText.setTextSize(12);
    paintText.setColor(Color.BLACK);

    Paint paint = new Paint();
    paint.setColor(Color.BRED);
    paint.setStyle(Paint.Style.FILL_AND_STROKE);
    paint.setStrokeWidth(1);
    Point lastpoint = null;
    for (int i = 0; i < mPoints.size(); i++) {
        Point point = new Point();
        mv.getProjection().toPixels(mPoints.get(i), point);
        x2 = point.x;
        y2 = point.y;
        if (i > 0) {
            canvas.drawLine(x1, y1, x2, y2, paint);
        }
        x1 = x2;
        y1 = y2;
        if (i == mPoints.size() - 1) {
            lastpoint = point;
        }
    }
    if (lastpoint != null) {
        canvas.drawText(mRoad.mDescription, lastpoint.x + 15, lastpoint.y + 25, paintText);
    }
}

@Override
public boolean draw(Canvas canvas, MapView mapView, boolean shadow, long when) {
    super.draw(canvas, mapView, shadow);

    GeoPoint da = null, db = null;
    if (directionPoint.size() > 0) {
        da = directionPoint.get(0);
    }
    if (directionPoint.size() > 1) {
        db = directionPoint.get(1);
    }

    // convert point to pixels
    Point screenPts = new Point();
    mapView.getProjection().toPixels(pointToDraw, screenPts);

    // add marker


Bitmap bmp = BitmapFactory.decodeResource(getResources(),
    R.drawable.red);
/*canvas.drawBitmap(bmp, screenPts.x, screenPts.y - 24, null);*/

bmp.recycle();

float rpixel = mapView.getProjection().metersToEquatorPixels(acc);
Paint paint = new Paint();
paint.setColor(Color.RED);
paint.setAlpha(150);
paint.setAntiAlias(true);
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeWidth(1);
canvas.drawCircle(screenPts.x, screenPts.y - 24, rpixel, paint);

paint = new Paint();
paint.setColor(Color.RED);
paint.setAlpha(50);
paint.setAntiAlias(true);
paint.setStyle(Paint.Style.FILL);
canvas.drawCircle(screenPts.x, screenPts.y - 24, 5, paint);

if (currentSelect != null) {
    screenPts = new Point();
    mapView.getProjection().toPixels(currentSelect, screenPts);

    // add marker
    bmp = BitmapFactory.decodeResource(getResources(),
        R.drawable.blue);
    canvas.drawBitmap(bmp, screenPts.x, screenPts.y - 24, null);
    bmp.recycle();
}

Integer uplat, uplong, botlat, botlong,
GeoPoint center = mapView.getMapCenter();
uplat = center.getLatitude() + mapView.getLatitudeSpan() / 2;
botlat = center.getLatitude() - mapView.getLatitudeSpan() / 2;
uplong = center.getLongitude() + mapView.getLongitudeSpan() / 2;
botlong = center.getLongitude() - mapView.getLongitudeSpan() / 2;
if (dbhHelper != null) {
    Cursor locCursor = dbHelper.fetchLocationByLatLng(upLat, upLong, botLat, botLong);
    if (locCursor == null) {
        Toast.makeText(getApplicationContext(), "Error in query database.",
            Toast.LENGTH_SHORT).show();
    } else {
        startManagingCursor(locCursor);
        if (locCursor.getCount() > 0) {
            paint = new Paint();
paint.setStrokeWidth(1);
paint.setARGB(255, 255, 255, 255);
paint.setStyle(Paint.Style.FILL);

            paint.setTextSize(12);
paint.setColor(Color.BLUE);
while (!locCursor.isAfterLast()) {
    String name = locCursor.getString(locCursor.getColumnIndex(  
        UKMapGuideDatabaseAdapter.LOCATION_COLUMN_NAME));
    Integer selLat = locCursor.getInt(locCursor.getColumnIndex(  
        UKMapGuideDatabaseAdapter.LOCATION_COLUMN_LAT));
}
Integer selLong = locCursor.getInt(locCursor.getColumnIndex(UKNMapGuideDatabaseAdapter.LOCATION_COLUMN_LONG));
GeoPoint p = new GeoPoint(selLat, selLong);
String directionpointname = "";
if (directionpoint.size() > 0) {
    GeoPoint pointA = directionpoint.get(0);
    if (pointA.getLatitudeE6() == p.getLatitudeE6() as pointA.
        getLongitudeE6() == p.getLongitudeE6()) {
        directionpointname = "A";
        da = null;
    }
}
if (directionpoint.size() > 1) {
    GeoPoint pointB = directionpoint.get(1);
    if (pointB.getLatitudeE6() == p.getLatitudeE6() as pointB.
        getLongitudeE6() == p.getLongitudeE6()) {
        directionpointname = "B";
        db = null;
    }
}
if (directionpointname.length() > 0) {
    name = name + " " + directionpointname,
}
screenPts = new Point();
mapView.getProjection().toPixels(p, screenPts);

// add marker
bmp = BitmapFactory.decodeResource(getResources(),
    R.drawable.blue);
canvas.drawBitmap(bmp, screenPts.x, screenPts.y - 24, null);
canvas.drawText(name, screenPts.x, screenPts.y + 16, paint);
locCursor.moveToNext();
locCursor.close();
// else {
//    stopManagingCursor(locCursor);
//    locCursor.close();
// }
}

if (searchResult != null && searchResult.size() > 0) {
    screenPts = new Point();

    // add marker
    bmp = BitmapFactory.decodeResource(getResources(),
        R.drawable.red);

    paint = new Paint();
paint.setStrokeWidth(1);
paint.setARGB(255, 255, 255, 255);
paint.setStyle(Paint.Style.FILL);
paint.setTextSize(12);
paint.setColor(Color.RED);
int i = 0;
for (GeoPoint p : searchResult) {
    mapView.getProjection().toPixels(p, screenPts);
    canvas.drawBitmap(bmp, screenPts.x, screenPts.y - 24, null);
    canvas.drawText(searchResultName.get(i), screenPts.x, screenPts.y + 15, paint);
    i++;
}
bmp.recycle();

if (da != null) {
    screenPts = new Point();
    mapView.getProjection().toPixels(da, screenPts);

    // add marker
    bmp = BitmapFactory.decodeResource(getResources(), R.drawable.blues);
    canvas.drawBitmap(bmp, screenPts.x, screenPts.y - 24, null);
    canvas.drawText("A", screenPts.x, screenPts.y + 15, paint);
    bmp.recycle();
}

if (db != null) {
    screenPts = new Point();
    mapView.getProjection().toPixels(db, screenPts);

    // add marker
    bmp = BitmapFactory.decodeResource(getResources(), R.drawable.blues);
    canvas.drawBitmap(bmp, screenPts.x, screenPts.y - 24, null);
    canvas.drawText("B", screenPts.x, screenPts.y + 15, paint);
    bmp.recycle();
}

drawPath(mapView, canvas);
return true;

private boolean isScrolling = false;
private float downX, downY, moveX, moveY;

@Override
public boolean onTouchEvent(MotionEvent event, MapView mapView) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        downX = event.getX();
        downY = event.getY();
        return false;
    }

    // == when user lifts finger ==
    if (MotionEvent.ACTION_MOVE == event.getAction()) {
        moveX = event.getX();
        moveY = event.getY();
        if (moveX - downX || moveY - downY)
            isScrolling = true;
        return false;
    }

    if (event.getAction() == MotionEvent.ACTION_UP && isScrolling) {
        isScrolling = false;
        return false;
    }

    if (event.getAction() == MotionEvent.ACTION_UP) {
        currentSelect = null;
    }
}
GeoPoint p = mapView.getProjection().fromPixels(event.getX(), event.getY());

Integer upLat, upLong, botLat, botLong;
GeoPoint center = p;
Point screenPts = new Point();
mapView.getProjection().toPixels(p, screenPts);
GeoPoint p = mapView.getProjection().fromPixels(screenPts.x + 20, screenPts.y + 20);
Integer delta = p.getLatitudeE6() - pl.getLatitudeE6();
upLat = center.getLatitudeE6() + delta;
botLat = center.getLatitudeE6() - delta;
upLong = center.getLongitudeE6();
botLong = center.getLongitudeE6() - 2 * delta;

Boolean selectAPoint = false;

if (dbhelper != null) {
    Cursor loccursor = dbhelper.fetchcocationnlatlonlong(uplat, uplong, botlat, botlong);
    if (locCursor != null) {
        Toast.makeText(getApplicationContext(), "Error in query database.", Toast.LENGTH_SHORT).show();
    } else {
        startManagingCursor(locCursor);
        if (locCursor.getCount() > 0) {
            m_button_regloc.setText("Edit");
            Integer selLat = locCursor.getInt(locCursor.getColumnIndex("UNMapGuideDatabaseAdapter.LOCATION_COUNMH_LAT"));
            Integer selLong = locCursor.getInt(locCursor.getColumnIndex("UNMapGuideDatabaseAdapter.LOCATION_COUNMH_LONG"));
            currentSelect = new GeoPoint(selLat, selLong);
            stopManagingCursor(locCursor);
            locCursor.close();
            selectAPoint = true;
        } else {
            stopManagingCursor(locCursor);
            locCursor.close();
        }
    }
}

if (!selectAPoint) {
    currentSelect = p;
    m_button_regloc.setText("Add");

    String loc = "Lat: " + (p.getLatitudeE6() / 1E6) + " - Lon: " + (p.getLongitudeE6() / 1E6);
    Toast.makeText(getBaseContext(), loc, Toast.LENGTH_LONG).show();
}

if (getdirectionflag == 1) {
    directionpoint.add(currentSelect);
    getdirectionflag = -2;
    m_button_direction.setText("Loc B");
} else if (getdirectionflag == -2) {
    directionpoint.add(currentSelect);
    getdirectionflag = -3;
}
m_button_direction.setText("Find route");
GeoPoint a = directionpoint.get(0);
GeoPoint b = directionpoint.get(1);
double fromLat = a.getLatitudeE6() / 1E6, fromLon = a.getLongitudeE6() / 1E6;
double toLat = b.getLatitudeE6() / 1E6, toLon = b.getLongitudeE6() / 1E6;
String url = RoadProvider.
    .getUrl(fromLat, fromLon, toLat, toLon);
InputStream is = getOnConnection(url);
try {
mPoints = new ArrayList<GeoPoint>();
    for (int i = 0; i < mRoad.mRoute.length; i++) {
        mPoints.add(new GeoPoint(
            (int) (mRoad.mRoute[i][1] * 1000000),
            (int) (mRoad.mRoute[i][0] * 1000000));
        )
        int moveToLat = (mPoints.get(0).getLatitudeE6()) + (mPoints.get(
            mPoints.size() - 2).getLatitudeE6()) - mPoints.get(0).
            getLatitudeE6()) / 2;
        int moveToLong = (mPoints.get(0).getLongitudeE6()) + (mPoints.
            get(mPoints.size() - 1).getLongitudeE6()) - mPoints.
            get(0).getLongitudeE6()) / 2;
        GeoPoint moveTo = new GeoPoint(moveToLat, moveToLong);
    }
    m_button_direction.setText("Direction");
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} return null;
private InputStream getOnConnection(String url) {
    InputStream is = null;
    try {
        HttpURLConnection conn = new URL(url).openConnection();
        is = conn.getInputStream();
        BufferedReader in = new BufferedReader(new InputStreamReader(is));
        String response = in.readLine();
        in.close();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } return is;
}
public class GeoUpdateHandler implements LocationListener {
    public void onLocationChanged(Location location) {
        if (location == null) return;
        int lat = (int) location.getLatitude() * 1E6;
        int lng = (int) location.getLongitude() * 1E6;
        GeoPoint point = new GeoPoint(lat, lng);
        // add marker
        MapOverlay mapOverlay = new MapOverlay();
        mapOverlay.setPointToDraw(point, location.getAccuracy());
        List<Overlay> listOfOverlays = mapView.getOverlays();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } return null;
    private InputStream getOnConnection(String url) {
        InputStream is = null;
        try {
            HttpURLConnection conn = new URL(url).openConnection();
            is = conn.getInputStream();
            BufferedReader in = new BufferedReader(new InputStreamReader(is));
            String response = in.readLine();
            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } return is;
    }
    public class GeoUpdateHandler implements LocationListener {
        public void onLocationChanged(Location location) {
            if (location == null) return;
            int lat = (int) location.getLatitude() * 1E6;
            int lng = (int) location.getLongitude() * 1E6;
            GeoPoint point = new GeoPoint(lat, lng);
            // add marker
            MapOverlay mapOverlay = new MapOverlay();
            mapOverlay.setPointToDraw(point, location.getAccuracy());
            List<Overlay> listOfOverlays = mapView.getOverlays();
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } return null;
        private InputStream getOnConnection(String url) {
            InputStream is = null;
            try {
                HttpURLConnection conn = new URL(url).openConnection();
                is = conn.getInputStream();
                BufferedReader in = new BufferedReader(new InputStreamReader(is));
                String response = in.readLine();
                in.close();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } return is;
        }
        public class GeoUpdateHandler implements LocationListener {
            public void onLocationChanged(Location location) {
                if (location == null) return;
                int lat = (int) location.getLatitude() * 1E6;
                int lng = (int) location.getLongitude() * 1E6;
                GeoPoint point = new GeoPoint(lat, lng);
                // add marker
                MapOverlay mapOverlay = new MapOverlay();
                mapOverlay.setPointToDraw(point, location.getAccuracy());
                List<Overlay> listOfOverlays = mapView.getOverlays();
                } catch (MalformedURLException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } return null;
            private InputStream getOnConnection(String url) {
                InputStream is = null;
                try {
                    HttpURLConnection conn = new URL(url).openConnection();
                    is = conn.getInputStream();
                    BufferedReader in = new BufferedReader(new InputStreamReader(is));
                    String response = in.readLine();
                    in.close();
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (MalformedURLException e) {
                    e.printStackTrace();
                } return is;
            }
        }
    }
    public class GeoUpdateHandler implements LocationListener {
        public void onLocationChanged(Location location) {
            if (location == null) return;
            int lat = (int) location.getLatitude() * 1E6;
            int lng = (int) location.getLongitude() * 1E6;
            GeoPoint point = new GeoPoint(lat, lng);
            // add marker
            MapOverlay mapOverlay = new MapOverlay();
            mapOverlay.setPointToDraw(point, location.getAccuracy());
            List<Overlay> listOfOverlays = mapView.getOverlays();
    }
```java
    listOfOverlays.clear();
    listOfOverlays.add(mapOverlay);

    mapView.invalidate();
}

public void onProviderDisabled(String provider) {
}

public void onProviderEnabled(String provider) {
}

public void onStatusChanged(String provider, int status, Bundle extras) {
}

public class GeoUpdateNoMoveHandler implements LocationListener {
    public void onLocationChanged(Location location) {
        if (location == null) return;
        int lat = (int) (location.getLatitude() * 1E6);
        int lng = (int) (location.getLongitude() * 1E6);
    }

    public void onProviderDisabled(String provider) {
    }

    public void onProviderEnabled(String provider) {
    }

    public void onStatusChanged(String provider, int status, Bundle extras) {
    }
}
```
Unit Test Code

```java
package uk.hw.ac.djk6.test;

import uk.hw.ac.djk6.UKMapGuideActivity;
import android.test.ActivityInstrumentationTestCase2;

import com.jayway.android roboumio solo.Solo;

public class MainTestCase extends ActivityInstrumentationTestCase2<UKMapGuideActivity> {

    private Solo solo;

    public MainTestCase() {
        super("uk.hw.ac.djk6", UKMapGuideActivity.class);
    }

    @Override
    public void setUp() throws Exception {
        super.setUp();
        //setUp() is run before a test case is started.
        //This is where the solo object is created.
        solo = new Solo(getInstrumentation(), getActivity());
    }

    @Override
    public void tearDown() throws Exception {
        //tearDown() is run after a test case has finished.
        //finishOpenedActivities() will finish all the activities that have been opened during the test execution.
        solo.finishOpenedActivities();
    }

    public void testClickMap() throws Exception {
        if (solo.searchButton("Map")) {
            solo.clickOnButton("Map");
            //Assert that NoteEditor activity is opened
            solo.assertCurrentActivity("Expected UKMapGuideActivity activity", "UKMapGuideActivity");
            solo.takeScreenshot();
            boolean expected = true;
            boolean actual = solo.searchButton("Sat");
            //Assert that Note 1 & Note 2 are found
            assertEquals("Click Map but not change to Sat", expected, actual);
            solo.clickOnButton("Sat");
            //Assert that NoteEditor activity is opened
            solo.assertCurrentActivity("Expected UKMapGuideActivity activity", "UKMapGuideActivity");
            solo.takeScreenshot();
            expected = true;
            actual = solo.searchButton("Map");
            //Assert that Note 1 & Note 2 are found
            assertEquals("Click Sat but not change to Map", expected, actual);
        } else {
            solo.clickOnButton("Sat");
            //Assert that NoteEditor activity is opened
            solo.assertCurrentActivity("Expected UKMapGuideActivity activity", "UKMapGuideActivity");
            solo.takeScreenshot();
            boolean expected = true;
            boolean actual = solo.searchButton("Map");
            //Assert that Note 1 & Note 2 are found
            assertEquals("Click Sat but not change to Map", expected, actual);
            solo.clickOnButton("Map");
            //Assert that NoteEditor activity is opened
        }
    }
}
```
```java
public void testClickAdd() throws Exception {
    solo.clickOnButton("Add");
    // Assert that NoteEditor activity is opened
    solo.assertCurrentActivity("Expected LocationActivity activity", "LocationActivity");
    solo.takeScreenshot();
    boolean expected = true;
    boolean actual = solo.searchButton("Register");
    assertEquals("Open Add location but not found Register button", expected, actual);
    actual = solo.searchButton("Back");
    assertEquals("Open Add location but not found Back button", expected, actual);
    solo.clickOnButton("Back");
    solo.assertCurrentActivity("Expected back to UKMapGuideActivity activity", "UKMapGuideActivity");
    solo.clickOnButton("Add");
    solo.enterText(0, "Loc");
    solo.text(1, "Desc");
    solo.clickOnButton("Register");
    solo.assertCurrentActivity("Expected back to UKMapGuideActivity activity", "UKMapGuideActivity");
    solo.clickOnButton("List");
    solo.assertCurrentActivity("Expected LocationListActivity activity", "LocationListActivity");
    actual = solo.searchText("Loc");
    assertEquals("Expected location named Loc is added", expected, actual);
    actual = solo.searchText("Desc");
    assertEquals("Expected location with description Desc is added", expected, actual);
}

public void testClickSearch() throws Exception {
    solo.clickOnButton("Search");
    solo.assertCurrentActivity("Expected SearchActivity activity", "SearchActivity");
    boolean expected = true;
    boolean actual = solo.searchButton("Search");
    assertEquals("Open Search location but not found Search button", expected, actual);
    actual = solo.searchButton("Back");
    assertEquals("Open Search location but not found Back button", expected, actual);
    solo.enterText(0, "London");
    solo.clickOnButton("Search");
    solo.assertCurrentActivity("Expected back to UKMapGuideActivity activity", "UKMapGuideActivity");
    solo.clickOnButton("Search");
    solo.assertCurrentActivity("Expected back to SearchActivity activity", "SearchActivity");
    solo.clickOnButton("Back");
    solo.assertCurrentActivity("Expected back to UKMapGuideActivity activity", "UKMapGuideActivity");
```