AN APPLICATION APPROACH TO MOBILE SIGNAL ANALYSIS

Submitted May 2013 in partial fulfilment of the conditions of the award of the degree of BSc (Hons) Computer Science.

Edward George
exg10u
School of Computer Science
University of Nottingham

Supervisor: Dr. Steven Bagley

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____/_____/_____

Acknowledgments

Many thanks to Dr. Steven Bagley for all his help and guidance in this project and for being a fantastic tutor and lecturer during the three years I've had at Nottingham. I have enjoyed every second of it!

A huge thank you to my forgiving friends, especially James. You guys always brighten up my day!

I also owe a special thank you to members and the executives of Comp Soc over the last three years especially Peter, Ben, Oliva and Miraj. Thank you all for the fantastic opportunities and the wonderful memories that followed.

I would like to thank my very patient girlfriend Chloe for her words of wisdom and help throughout this year and this project. She always knew what to say when I needed it most.

I would also like to thank my wonderful Grandparents and all of my extended family for all their love, support and kindness during this dissertation. It was much appreciated.

Finally I would like to thank my loving parents, Tracy and Roger, as well as my sister Amelia for all their support during my time at Nottingham and *especially* during this project. Without your encouragement and belief in me, I could not have got this far. I am truly blessed to have you as my family.

(Plus your suggestions weren't always as crazy as they sounded!)

Abstract

This dissertation aims to explore the creation of a mobile phone application that will enable its users to gain a better understanding of their current location's signal properties, namely its data speed or current strength, and to evaluate how effective the application is in doing so. A number of interfaces are proposed, discussed and evaluated in addition to the final proposal.

The research looks at how possible it is to use the application globally and not be limited to network configurations as found in the United Kingdom, as it is widely regarded that the UK's mobile network is modern and well managed.

Existing systems that provide a similar service are discussed. These systems are compared, evaluated and used in the justification of the proposed application.

The design of the chosen interface is detailed and a software requirements specification for the app is stated.

The implementation of the application is fully detailed with all key decisions being discussed and justified.

Contents

List of Tables

List of Figures

1

Introduction

Mobile telephones have become a necessity in modern life, with their use in everyday activities allowing for the rapid pace of society and access to information across the globe *'regardless of location'*. However, in this modern era, there are *still* mobile signal 'black spots', areas of poor coverage, that can cause disruption to users of some of the largest networks!

With over 6 billion mobile subscriptions globally [1], mobile infrastructure is a huge requirement of today's society. But it can often be the case that 2G and 3G signal strength and speed can vary significantly in even the most developed countries.

The research looks at how a mobile application can be used to give mobile phone users a better understanding of their geographical location's real-time mobile signal strength, its speed and also help to locate so called 'hot spots' and 'black spots', areas where mobile signal is either seemingly in abundance or non-existent.

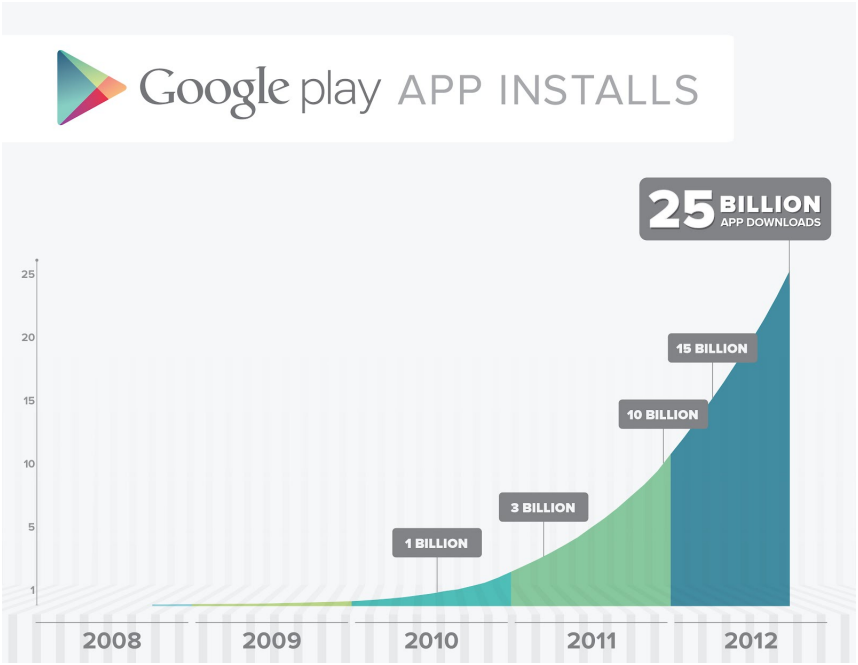
1.1 Motivation

As an avid smart-phone user and enthusiast, it is often frustrating to find myself struggling to access mobile data signal or standard network coverage. It can be difficult to locate an area where signal is present and more often than not, causes me to consider my smart-phone 'useless' in certain locations.

I would like to be able to provide people with more detailed information about their mobile phone's signal strength and speed in the hope this will enable people to use their phone more productively. By using the information my solution provides, I aim to make phone users perform more informed decisions on how and when they should use their phone.

I am also interested in the design of mobile applications for popular platforms

such as Apple's iOS and Google's Android platform. Since beginning university, the leading app stores for these platforms (Apple's 'App Store' and Google's 'Google Play Store') have both topped 25 billion downloads [2][3], with the Google Play Store's downloads increasing by 2500% in the last 3 years.



Android App Downloads [4]

From this trend, it can be assumed that many smart-phone users would find an application approach to be the most effective form of solution and consider an application that could provide information, such as that previously described, to be a vital asset to them and I am interested to see if this is the case once the solution is developed.

1.2 Aims and Objectives

This study aimed to carry out the fundamental research and development of mobile technology that would effectively relay mobile signal information to the user in real-time, as well as aiming to determine how successful and useful the system would be. The study aimed to research, discuss, implement and evaluate a mobile phone application to complete this.

This study proposed to carry out the relevant research into developing a basic implementation of an 'Android application' with a number of interfaces and methods of signal analysis having been implemented and evaluated. A final running version of the system has been tested and the solution evaluated.

The project's key aims and objectives have been formulated and are listed below:

Understanding Mobile Signal Technologies

To develop an application to accurately measure various aspects of a mobile device's received signals, it is important to understand and research the fundamental theory behind the technologies. It is also important to understand how networks vary globally, as well as within the UK, as this data would be useful in determining the applications potential user base.

As different mobile signal technologies exist worldwide [6], it may be the case that this research should be limited to the UK and the technologies we use as standard, to avoid over complicating the project given the time constraints upon it.

Understanding Signal Strength Role In Phone Usage

To develop a successful application that potential users will view as 'useful' it will also be important to determine *why* people require a strong signal strength, besides the obvious standard phone usage of making/receiving calls or SMS messages.

This will be beneficial as it will be possible to pinpoint exactly *how* to tailor the app for the requirements of its users and what information they may want my application to display.

Understanding Signal Speed Role In Phone Usage

Again, it will also be important to discover exactly how potential users use their devices in the context of signal speed. Why do users require a fast signal? What speed is ideal and how rare is it to get the desired speed?

The project looks to answer these questions and use the results to better the final application.

Developing for the Android Platform

Android is currently the world's most popular mobile platform [7] and has an extensive application programming interface (API) that uses Java as the default language [9]. The aim is to create an application that uses recent features of the latest Android API to create an application that will both engage the user and also effectively convey the information the application is designed to display. To complete this, programming for Android will have to be learnt during the duration of the project.

2

Background

This section aims to explore the background information relevant to completing the project. This includes a discussion of mobile network technologies and a discussion on causes of signal 'black spots'.

2.1 Mobile Signal Technologies

Mobile telecommunication protocols have differed globally since their creation nearly 35 years ago. Initially, the signals sent between base stations and phone receivers was an analogue signal and prone to interference. These first generation (1G) protocols were soon scrapped in favour of second generation (2G) digital protocols such as the 'Groupe Speciale Mobile', later renamed 'Global System for Mobile Communications' (GSM).

Released in Europe in the early 90's, GSM sees global use in a range of frequencies (900 - 1900MHz) and is widely used for text messaging and call capabilities synonymous with early mobile phone usage. The protocol hit over 6 billion connections since its creation in 2011 and continues to service 75% of the worldwide mobile network market [10].

As the internet grew exponentially larger in the late 90's and early 00's, the data systems implemented at the time were not fast enough to reliably browse the internet. Protocols such as the 'General Packet Radio System' (GPRS) and 'Enhanced Data Rates for Global Evolution' (EDGE) were adopted to fill the void and were commonly known as 2.5G technologies as they were only additional protocol layers added to the then current 2G system.

To provide widespread mobile internet access, 3G services were launched in the UK on 3/3/2003 and used the 'Universal Mobile Telecommunications System' (UMTS) protocol that could allow devices to have a peak data rate of around 2MB per second. Other countries such as America opted for a 'Code division multiple access' (CDMA2000) protocol allowing data rates of up to 3.1Mb per

second. [11]

Forth Generation (4G) networks were released in the UK on the 21st August 2012 and aim to provide speeds upwards of 20Mbps. This technology will not be discussed within this dissertation due to it's current limited availability [12].

2.2 Cause of 'Black spots'

In the UK it is widely acknowledged that the 2G, non-mobile data coverage is 'good' but in the last couple of years some mobile providers still actively have as much as 20% of the UK that is not covered by their 3G networks [13] - an area covering 49,000 km^2, roughly the size of Slovakia [14].
Such Mobile phone black spots occur for numerous reasons and tend to fall into two categories of 'man-made' and 'natural' causes.

2.2.1 Human Factors

'Human factors' refer to man-made instances that cause mobile signal to be lost in certain areas. These factors are often avoidable and usually the results of poor planning. Some of these factors are listed below.

- **Lack of Infrastructure**

  Globally, black spots of certain signal types (2G/3G) can still frequently be attributed to a lack of infrastructure to cover an area. Important infrastructure such as mobile signal masts can have a range from 1-2 miles up to around 20-35 miles depending on the surrounding terrain, meaning to cover a large area tens of thousands of masts would be required.

  The infrastructure is expensive to implement and maintain, meaning many poorer countries still have limited 2G and 3G coverage, such as Kenya where 3G coverage is currently available in only 3 major cities [15], representing a fraction ($\approx 9\%$) of the overall country's population [16][17].

- **Technical Fault**

  Mobile networks are under constant use 365 days a year and can occasionally fail, causing either little to no disruption or a widespread issue that causes loss of signal across a large geographical area.
  An example of an outage causing widespread blackouts occurred both in October 2012 when around 2 million phone users in the UK were left with 'intermittent service' [18] and July 2012 when 'hundreds of thousands' of people were left without call, SMS messaging (2G) or internet (3G) capabilities [19].

Faults can either be accidental or an act of sabotage but in well established networks, both are fairly infrequent and unlikely to effect large volumes of users.

- **Other**

  There are other human based factors that can weaken signal in certain areas.

  Construction of large buildings can change a networks layout by blocking signals, causing a possible loss of service to those in the building's immediate shadow. Similarly, the same effect can be observed in large tunnels as signals cannot reach the phone's receiver due to being unable to penetrate rock or propagate the length of the tunnel through the openings.

2.2.2 Natural Factors

'Natural factors' refer to natural phenomena and properties of electromagnetic waves that can cause areas where signals cannot be received. Some of these are described below.

- **Hills and Uneven Terrain**

  The geographical location and layout of a mobile phone transmitter's coverage area can directly effect the strength of signal received by a device within it. Signals propagate in an infinite ellipsoid radiation pattern emitted from a transmitter which forms what is known as the 'Fresnel Zone'.

  The radiation pattern forms waves that become either in sync and amplified (in phase) causing a strong signal or cancel out (out-of-phase) causing a weak signal. This phenomenon known as phase difference and is caused by the alignment of two or more waves, causing the overall resulting wave to change in amplitude.

  To maximize the receiver strength, the effect of the out-of-phase signals must be minimized by removing obstacles from the radio frequency line of sight. The strongest signals are on the direct line between transmitter and receiver and always lie in the first Fresnel zone. Figure 2.1 shows a graphical representation of a Fresnel zone and highlights the 'first' Fresnel zone.

  Using the formula:

$$F_n = \sqrt{\frac{n\lambda d_1 d_2}{d_1 + d_2}}$$

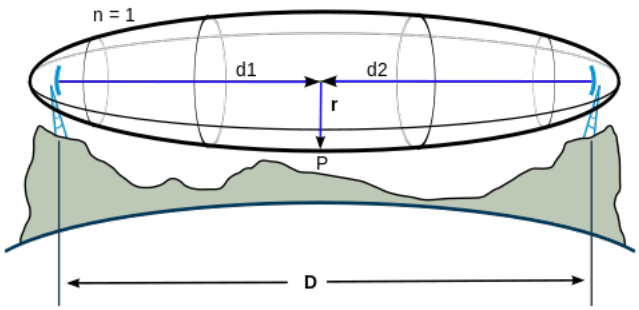


Figure 2.1: **Fresnel Zone [20]**

It is possible to calculate the radius of the nth Fresnel Zone (F_n) radius in metres at any point P using the distance from each antenna (d_1 and d_2) and the wavelength of the transmitted signal (λ).

This is important as any obstructions in the odd numbered Fresnel zones will create signals with phase shift of 0 to 180 degrees. Within the even zones they will cause transmitted signals to be 180 to 360 degrees out of phase causing even numbered zones to have the maximum phase cancelling effect on any received signals. Conversely, odd numbered zones may received 'boosted' coverage through multiple signals becoming in-phase [21].

- **The Properties of Radio Waves**

  The properties and principles of radio waves used by mobile phone networks can be used to explain why signal becomes weaker in certain areas.

  Radio waves propagate from a transmitter much like the ripples from a dropped stone in a pond. They travel out radially, progressively covering a wide area but losing strength as they do so. This action can be generalised as it is true that the area of the surface covered is proportional to the radius (r) and the signal strength is inversely proportional to the distance from the source to the receiver squared (d).

$$A \propto r$$

$$S \propto \frac{1}{d^2}$$

  However, this is only true for areas where there is no interference from obstacles such as those found frequently in 'terrestrial settings'. Buildings, large trees and alike can all contribute into causing the signal to weaken quicker. In some cases, the signal can weaken to a factor of d^{-4}. [22]

$$S \propto \frac{1}{d^4}$$

For maximum efficiency, it would be ideal if mobile base stations could cover an area in a tessellating pattern such as that of a hexagon. However, in reality the

Figure 2.2: **Ideal layout of Mobile Transmitters [23]**

area covered is radial and therefore prone to 'coverage gaps' should the coverage of multiple sites not overlap correctly.

Another property of waves is their ability to reflect, refract and diffract upon impact with certain obstacles causing the signal path to change between the base station transmitter and a phone's receiver.

The reflection of a signal works in exactly the same way for any electromagnetic wave. The incoming wave, known as the incident wave, strikes the surface and reflects. The angle between this wave and the perpendicular to the surface is the 'angle of incidence' θ_i that is also equal to the angle of reflection θ_r as seen in Figure 2.3. Reflection can cause signal loss due to the absorption of some of the signal by the medium it reflects off. [22]

Refraction of a mobile signal also acts in the same way it does for natural light[1]. The phenomenon is caused by a wave passing through a material with a differing 'refractive index' to that of the previous material[2]. This changes the path of the wave as demonstrated by Figure 2.4. This change of direction can cause signals to miss an area within their line of sight and cause an area in which signal is lost.

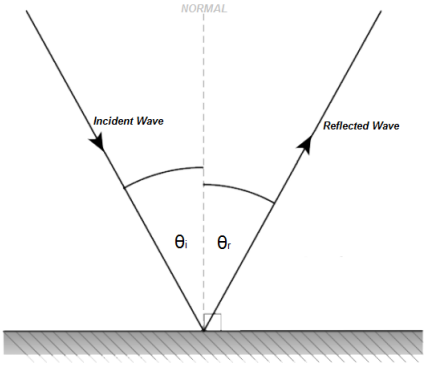


Figure 2.3: **Reflection of Electromagnetic Waves**

$$\boxed{\theta_i = \theta_r}$$

[1] Such as the 'bent' appearance of a drinking straw in a glass of water
[2] e.g. 'free air' with a refractive index of ≈1, depending on the atmospheric conditions

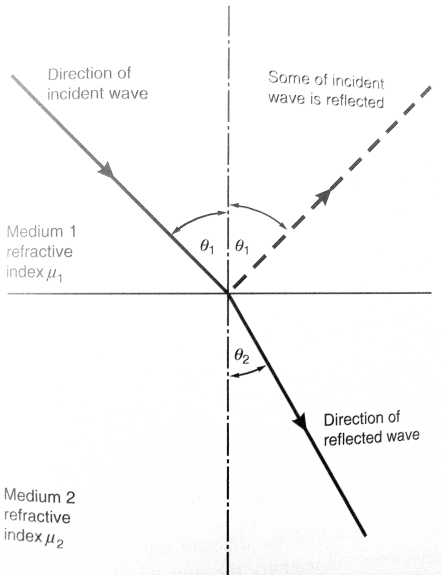


Figure 2.4: **Refraction of Electromagnetic Waves**

$$\boxed{\mu_1 \sin\theta_1 = \mu_2 \sin\theta_2}$$

2.3 Conclusion

Through my research into the potential causes of signal loss from a mobile base station to a mobile phone's receiver, it is clear that there are many mitigating circumstances that can lead to signals not reaching the device, causing possible areas of low signal strength know as black spots.

This infers that when networks are planned, rigorous analysis of the geographic locations must be undertaken to ensure base stations are placed appropriately with the aim to provide the strongest possible service to all. However, sudden changes in the environment, such as the construction of (large) buildings in metropolitan areas, can mean an area's signal strength distribution is changed and new patches of poor signal are created.

This would suggest that both emerging economic countries with small networks and larger more established economic countries with large networks, face a constant challenge to provide the best signal to all whilst the consistency of signal in an area changes.

This hypothesis is what leads me to believe that an application for smart-phone users to measure signal strength and speed in their current location may be a popular tool that could be widely used to help gain a better understanding of an area's signal speed/strength.

3

Relevant Existing Systems

This section briefly explores and describes relevant current systems in place that allow users to view mobile signal strength/speed or other mobile telecommunication data in their current local geographical area. These systems fall into two categories of either providing 'live' data or 'historic' data.

Live data systems can actively determine an area's signal properties in *real-time* or in a relatively recent time-frame, allowing users up-to-date information and accuracy.

Historic data systems provide an overview of an area's perceived signal properties, commonly over a large area, that can be used as a general reference. These systems are usually updated sporadically as their purpose is often to serve as a 'snapshot' of the area in question at the previous given time of data collection.

3.1 Live Systems

The following systems provide users *with* or *collect* real-time network system information.

3.1.1 OpenSignal Application

OpenSignal is a mobile application for Android and iOS provided by 'OpenSignal.com'. The application, created in late 2012, aims to provide users with information about how to "get better signal, find nearby Wi-Fi networks and keep track of your (data) usage" [24].

The application has multiple sections, categorised by the information they portray to the user. This includes sections for Wi-Fi information, network coverage maps, usage statistics and data speed tests. The information provided

includes the strength of signals in decibels as well as absolute strength units.

The application uses 'touch' as the main source of user interaction, meaning navigation round the app is seamless and quick. The tabbed interface allows a quick change between different screens through a sliding touch in the direction of the tab required or through touching the tab requested.

Figure 3.1: **An example of the interface provided by the OpenSignal app [24]**

Finally, the application allows users to add to the company's database by reporting any issues with reception at the user's current location.

The application is used by over a million user's worldwide and is currently the most popular app of its kind for the Android market [25].

3.1.2 openBmap Application

The 'openBmap' application, initially released in 2009, is an Android and Windows Mobile app created for the purpose of producing of 'the world's largest free and open map of wireless communicating objects', including both cellular and Wi-Fi data. The database, at present, has data on 194 countries and over 600 mobile networks [26].

The app provides a very simplistic interface that allows the background 'recording' of signal data that can then be used to contribute to the online database. This is shown in Figure 3.2. There is no 'live view' to allow the user to view the current signal as this is completed in the background of the application but access to the database produced by openBmap is available as an online API.

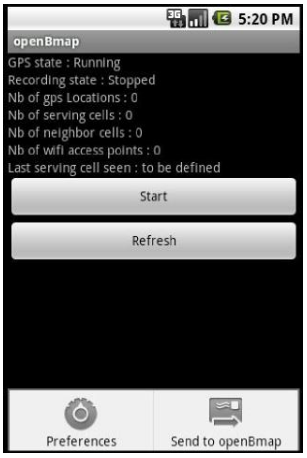


Figure 3.2: **An example of the interface provided by the openBmap app**

3.1.3 Mobile Phone Operating System

The operating system of a mobile phone is always actively monitoring the current network status. It is common for this to be displayed in the form of a maximum of four or five bars, depending on the manufacturer of the device (Figure 3.3).

Figure 3.3: **The classic representation of mobile signal strength**

There is currently no industry standard indicating what a 'bar' *means*. This infers that comparing the number of bars between phones of different models, even if they are produced by the same manufacturer, it does not necessarily mean they represent the same signal strength despite being visually similar (Figure 3.4).

It is very uncommon for manufacturers to show actual signal strength, measured in decibel meters (dBm) despite this unit of measurement having been the industry standard since it was proposed in 1940 [29] and through its adoption by manufacturers in the early days of mobile telecommunications. This absolute strength value remains hidden unless specifically requested on most current Mobile operating systems:

- Android: Under the phone's settings, absolute strength can be viewed but not constantly displayed.
- iPhone: Dialling ***3001#12345#*** and pressing call will display the absolute strength in the place of the standard bars [30].

It is critical, from a manufacturer's perspective, that the algorithm used by the operating system to display these signal bars is thoroughly tested otherwise

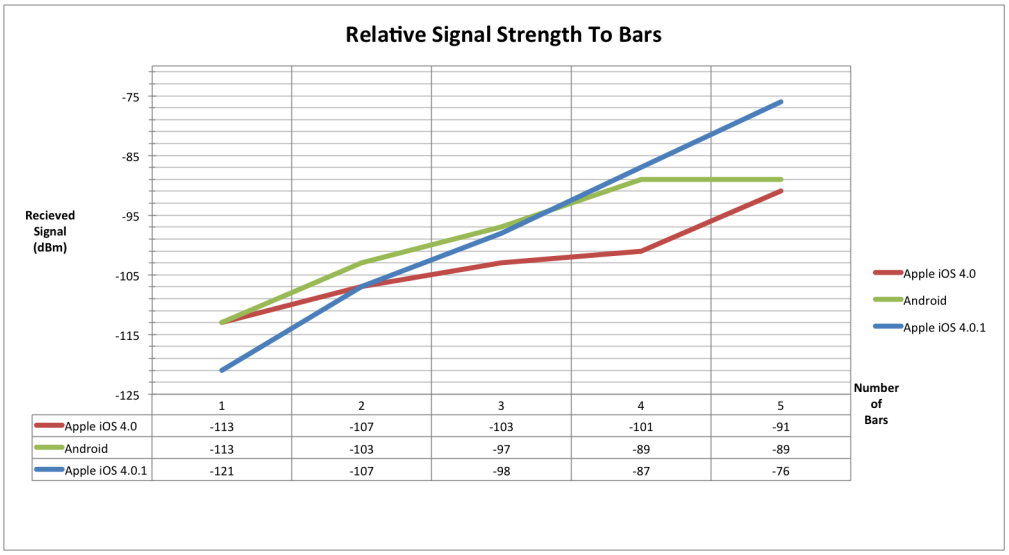


Figure 3.4: **A graph to show the differences in Android and iOS, showing the same number of bars [27] [28]**

incidents, such as the iPhone 4's initial negative press [31], can cause user confusion and damage a company's reputation through displaying incorrect signal data in the form of bars.

3.2 Historic System

The following systems provide details about a geographical locations mobile signal properties much like the previous systems described. However, these systems do not provide or collect data in real-time and are therefore categorised as 'Historic'.

3.2.1 Opensignal.com

Founded in 2012, Opensignal.com is a website that allows users access to a global map of 2G and 3G coverage through a Google Maps interface [32]. Users can enter a postcode, select a country or interact with the map directly to view an area's signal strength in the form of a coloured 'heatmap', in which different colours represent an areas strength (blue representing weak signal and red representing a strong signal).

The website also details the apparent locations of radio base stations with information about their operator as pinned points on the map interface (Figure 3.5).

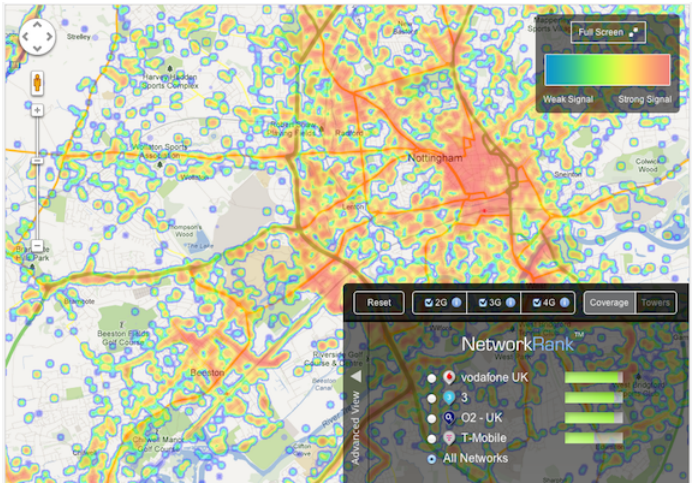


Figure 3.5: **An example of the interface provided by OpenSignal.com**

3.2.2 SiteFinder - Ofcom

Within the UK, The Office of Communications (Ofcom) is the government body responsible for regulating telecommunications. Sitefinder, an Ofcom run website, offers a map interface to locate mobile base stations within the United Kingdom. The website states that "Sitefinder was set up as a result of recommendations of the Stewart Report in 2000. It is a voluntary scheme under which mobile network operators make information available on the location and operating characteristics of individual base stations, so that people who wish to inform themselves about this can do so." [33]

The website also offers links to the dataset containing information about the base stations [34]. This dataset includes the station operator, an Ordinance Survey grid reference for the location, the frequency used by the antenna and other information about the site. However, some mobile operators have recently refused participation and therefore the dataset is not 100% accurate as of April 2013.

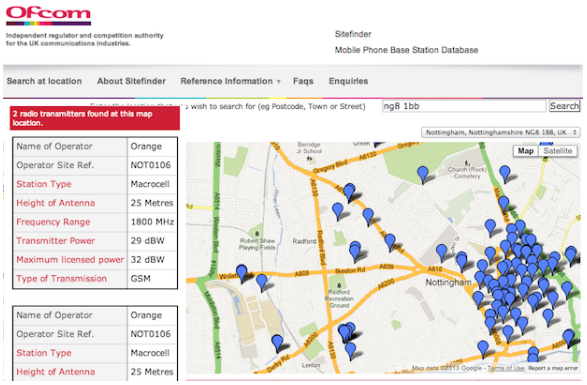


Figure 3.6: **An example of the interface provided by Ofcom**

3.2.3 Operator's Websites

Globally, a recent survey estimates there to be around 800 different mobile network operators [35]. Many of these network operators provide detailed maps of their perceived coverage for customer use in the form of interactive maps on websites. These maps are used to show the service the company provides over a large geographical area. This is mainly used as a customer service tool to allow users to see if the company provides to an area the customer requires.
Many companies use this interface as a platform for customers to make maintenance requests or complaints about the signal they are experiencing in the area.

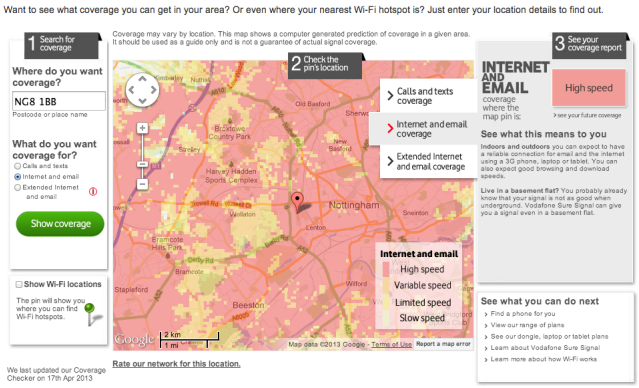


Figure 3.7: **An example of the interface provided by Vodaphone UK**

4

The Proposed Solution

This section aims to evaluate the existing systems, their setbacks and discuss the overall issue the research intends to solve.

As discussed earlier in the Background Section 2.2, there are numerous cases in which loss of signal and black spots can be caused due to human or natural factors. This research has led to the proposition and development of a mobile application that can accurately display live data to the user on their device's signal properties. The initial development of the conceptualised application was completed with the following presumption for it's use: A user is in an area of unfamiliarity and requires a strong mobile signal. This could be for professional reasons, i.e. checking e-mail, or for purely leisure purposes, e.g. social media. The user can use the app proposed to find information on how quick/strong the connection is currently and use this information to determine what their phone usage should be at that given location. Should the user be on the move, the application can track them and provide information on previous location's signal.

The existing systems described in Chapter 3 do not complete this initial concept proposed for my applications.

The OpenSignal application for Android and iOS, whilst mostly displaying an excellent user interface, displays it's key **live** signal information over many pages and not in a centralised location. It should also be noted that, by default, the application does not provide live signal data in terms of strength units (dBm), but in the form of bars. The system provides plenty of features but *could* be seen as being distracted from one of this research's key goals of providing live data effectively. The heat-map interface, despite locating the user's current location, does not provide live data; something the proposed application should aim to achieve. This consequently means the system provided by the OpenSignal application is not satisfactory in meeting the project's aims.

The openBmap application collects live signal data, but currently has no interface for viewing this collected data, as it is sent to the database in the

background of the app. This is something the proposed design must avoid, as the aim is that the visualization of this data should inform the user and consequently must be visible on request and subsequently, a system such as openBmap, does not meet this aim. The large database and API openBmap provides to access it, may be of some use for the project as previous recorded data could potentially be accessed to compliment the live data the project's app intends to display.

The mobile phone operating system is the only source from where the signal information can be extracted through purely programmatic means, i.e. no physical hardware, additions or changes. It is therefore a vital resource for the creation of the project. The current method of display adopted by most, if not all manufacturers of signal 'bars', has been proved to be largely complicated and essentially meaningless in Section 3.1.3, and this is a key area that the project aims to address.

The OpenSignal.com website provides an excellent interface for users to find out a location's signal and is not limited to a single network or country as it has a search feature allowing users to find information about a desired location. The website does, however, only provide data in the form of a heat map that is static meaning the heat-map may be updated but the data is not 'live'. The website can be viewed from a mobile device and displays a map view with mobile base stations placed in their geographical location along with information of the company in charge of maintaining the site. Unfortunately there is little that this system does to complete the aims and objectives of the project's proposed application in terms of live data. In terms of interface design, the map interface may be worth considering during the design stage of development for it's ease of use.

Ofcom's Sitefinder website has a large amount of information about the base stations used within the United Kingdom, but only has information of this nature. There are no live base station readings or statistics. The web interface is similar to those using map views, i.e. OpenSignal.com and the OpenSignal app but provides information about base stations and not the network coverage of an area. The dataset, provided by Ofcom, of these base stations could potentially be used within the solution despite being 'out of date'. It could add useful information about an area and serve to further enhance a user's understanding of the area's signal speed and strength. However, the dataset, currently provided in the form of a Microsoft Excel spreadsheet, would require much work before it would be usable due to the large volume of data present[1] and the number of fields for each record. Due to it's size, it would almost certainly not be prudent to store such a database locally on a device, meaning another method of usage should be adopted.

Finally, mobile network operator websites provide a very similar interface to that of OpenSignal's website, offering a heat map of the specific provider's coverage

[1] Currently the dataset contains ≈145,000 records

Table 4.1: **A comparison of the six relevant and existing systems**

	OpenSignal	**openBmap**	**Mobile OS**
Live Signal Speed	✔	✗	✔
Live Signal Strength (dBm)	(Not by default)	✗	✔
Other Signal Info	✔	✗	✔
	OpenSignal.com	**Sitefinder**	**Operator's Site**
Live Signal Speed	✗	✗	✗
Live Signal Strength (dBm)	✗	✗	✗
Other Signal Info	✔	✔	✔

Table 4.1 can be used to quickly compare the six relevant current systems as described in Chapter 3 as well as view what relevant features the system offers.

for a region and allowing users to search for a location to view the coverage there.

4.1 Summary

From the table and the comparison of the existing systems it seems that using the phone's operating system is the most prudent way of completing the project as it has access to the key signal data aspects the application aims to portray to the user. No other single system provides all the information required apart from the OpenSignal app, despite not showing actual signal strength by default.
This would suggest that the project's outcome may be a unique application and fit this gap in the market.

5

Design

5.1 Introduction

This chapter outlines the design of the proposed application and its features as a comprehensive description of the design chosen and how it addresses the problem to hand. It also contains the system specification and different interfaces available for development.

5.2 Software Requirements Specification

A Software Requirements Specification (SRS) is a specification to determine and describe the complete behaviour of a piece of software that is to be developed. This is useful as it can give a good overview as to whether a piece of software has been completed to the necessary standard by testing against the SRS during development and the tested phase.

This SRS will be using an adapted version of IEEE standard 830 to format the document in the correct manner and with relevant information [36].

5.2.1 Scope

The scope of the software to be produced is as follows:

- The software to be produced will be in the form of a Mobile Phone Application
- The Android platform will be used to produce this application
- The application to be produced is to be called 'Signal Analysis Tests' (Working Title)

- The application will provide its users information about their devices current signal's properties
- 'Signal Analysis Tests' aims to inform the user and lead them to make better choices in their phone's usage depending on the information presented to them by the application
- 'Signal Analysis Tests' also aims to have potential use globally on different mobile protocols

Definitions, Acronyms and Abbreviations

- From here on within the SRS, 'the application' or 'app' refers directly to the proposed application 'Signal Analysis Tests' unless explicitly mentioned
- 'Mobile signal' refers to the signal a mobile device receives from either a 2G or 3G network connection under the protocols described in Section 2.1
- The acronym 'GPS' refers to the Global Positioning System, a satellite navigation system that can be used by mobile devices to track a user's current location
- 'dBm' refers to Decibel meters, the unit associated with signal strength in telecommunications
- 'Mbs' refers to 'Megabits per second', the unit commonly associated with speed over a mobile network[1]

5.2.2 Overall Description

This section of the SRS is to describe the factors that directly affect the software to be developed and it's requirements.

Software Interfaces

- Any mobile device that wishes to use the app should be running a minimum of Android's 4.0.3 'Ice Cream Sandwich' release as, at the time of development, this is the latest SDK available for developers and will be the minimum platform the app is developed for
- Any device should also contain the default Google libraries already installed
- Using third party tools, the app will be loaded onto a device. The Google Play app store will not be used

[1]Not to be confused with Megabytes per second.

Memory Constraints

- Any mobile device that wishes to use the app should have between 1MB and 5MB of free storage for the application to be installed onto the device. The app aims to fall roughly around 1.5MB in size, but any changes can be accounted for with this constraint
- The device should have 512MB of RAM or greater. This is to ensure the app can run smoothly

User Characteristics

- The application aims to be used by 'keen smart phone users and enthusiasts'. Therefore it can be expected that many users of the app will intuitively know how to use the app for their own benefit as they have a good level of understanding how mobile applications tend to work. Nevertheless, it should be the aim of the software to provide a good and clear user experience *regardless* of their perceived skill in working with mobile phone applications, despite the app not necessarily being marketed towards these individuals

Constraints

Constraints refer to any item that may be deemed to limit the options the project can undertake during development.

- **Time**: The project has to be completed no later than the 8th May 2013 and is non-negotiable.
- **Privacy**: The project has to comply to regulations and good practise regarding the private data of any user, should it be collected. This includes the storing of a user's location
- **Programming Ability**: The project is to be completed in Android in the hope it produces a functional app but also as tool for learning, therefore the ability of the programmer should be taken into account during the design and implementation processes
- **Relevant Tools and their Availability**: The project is to be completed on an Android 4.0.3+ device. It is currently unknown if one of these devices is available for the project should the current test phone become unavailable. This may mean specifications are required to change should this occur
- **Cost**: The project must not have any unjustified or excessively large costs relating to the project during the whole life cycle of it's development

5.2.3 Specific Requirements

Functions

These *functional requirements* define the **key and fundamental features** that the software must implement, especially during input and processing/providing output.

- **F1:** The system shall display the current signal strength of the device (dBm)
- **F2:** The system shall display the current speed of signal of the device (Mbs^{-1})
- **F3:** The system shall handle user input in the form of 'touch' on the device's touch screen
- **F4:** The system shall track the user's current location using GPS
- **F5:** The system shall handle any errors accordingly

The following sub sections describe the *non-functional* requirements. These 'quality attributes' refer to the aspects of how a system is designed and implemented. It does **not** refer to what **functions** the system aims to provide.

Appearance

- **A1:** The system must 'look professional'
- **A2:** The system's layout must be intuitive and easy for users to understand and use
- **A3:** The system must use a Map layout to display location information
- **A4:** Any compatible 4.0.3 Android phone should have a resolution of at least 320 by 480

Performance

- **P1:** The system should, once connected to GPS, accurately locate the user
- **P2:** The system should prompt the user to enable GPS should the device support it and it be disabled

- **P3:** The system should, after locating the user, accurately track the user to a high degree of accuracy (between 0m - 150m)
- **P4:** The system should collect signal strength to an appropriate degree of accuracy and precision
- **P5:** The system should collect signal speed to an appropriate degree of accuracy and precision
- **P6:** The system should have a maximum reaction time to user input of 3 seconds
- **P7:** Any actions that may require a wait longer than 3 seconds must inform the user of the process' progress to completion

Documentation

- **D1:** The system should require little to no documentation for it to be correctly used by users

---

5.3 Initial User Interface Design Proposals

After carefully analysing the SRS, some initial mock-up designs for the application were produced using wire frames software. The following interfaces and views were explored:

5.3.1 Map View

It was decided that the application will use a map interface, building upon similar designs from systems described in Section 3. This is a tried and tested method and arguably the best visualization tool available for this kind of application as it allows users to view accurately the surrounding geographical location through built in tools the interface supplies. Many map APIs, including Google Maps and the default Android map, also allow 'overlays'. These overlays, often in the form of graphics, allow developers to add layers over the map view to display information. This will allow relevant signal information to be added to the screen at the current location of the user.

Map View Proposal 1

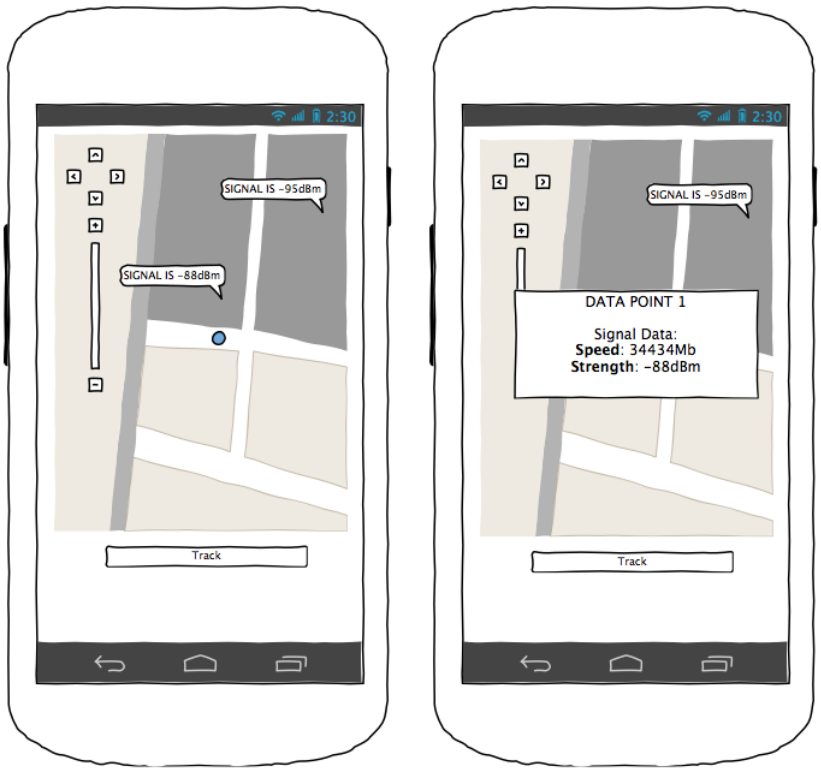


Figure 5.1: **Example 1 of potential Map View design**

This was the first design that was initially considered. It features a simple interface of a single map view and button to track the user via GPS. The idea behind this was to not over complicate the interface and to focus on the main system specification and it's requirements.

Once GPS had located the user, the map view would display overlays in the form of speech bubbles containing signal information for that location. Clicking on these bubble would bring up a dialogue box containing further information about that particular point such as the longitude and latitude of that point as well as the signal strength experienced at that location.

The track button would be used to request Android's GPS service to accurately track the user's current location.

The map view would be navigable through the use of touch and *gestures*. This includes a 'pinch' motion to zoom the map in and out.

Map View Proposal 2

Figure 5.2: **Example 2 of potential Map View design**

This design differs slightly from the first documented in Figure 5.1. It still uses a Map View with all the features as previously described including bubble overlays, gestures and GPS tracking. The main differences with this design is the use of a graph to display real-time signal data to show the user live statistics whilst also tracking location and providing locational based information.

There are a number of libraries available that allow for graphs to be created easily in Android and these will be discussed and evaluated later in this document.

This design, despite being harder to implement than the first design, seems to be a unique idea that was not present in any of the current systems available and can therefore be seen as a desirable feature that would add a 'unique selling point' to the application. It was for this reason, this design was taken forward from the initial design stage and implemented.

5.3.2 Speed Test View

The Speed Test view initial designs were also designed based on ideas adapted from previous system's designs but it was determined that it was also important to try to keep the design 'original'. This led to a further two proposed ideas:

Speed Test View Proposal 1

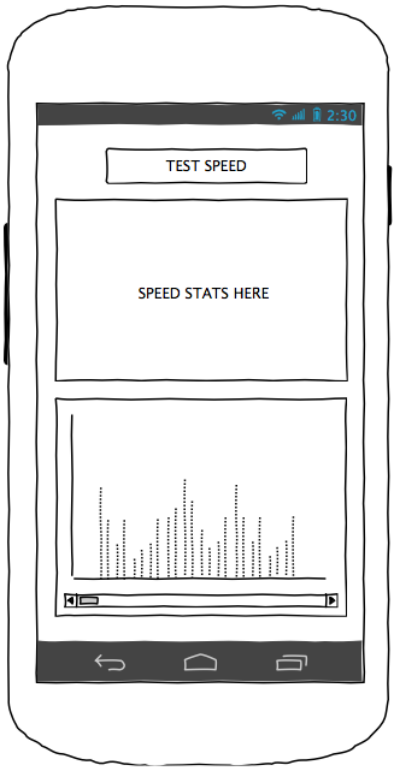


Figure 5.3: **Example 1 of potential Speed Test design**

The first design was designed to be simple. This design was created with full focus on the specification and the requirements it set out.

The button at the top of the screen allows users to partake in a speed test after clicking. The design, once again, uses a graph to show live speed information once the user has opted to take a speed test. This graph could be used to show any changes in speed during the download period or to compare previous results with the then current test data. Above this is a text view that would contain data about the latest speed test taken, including maximum speed, time to download and average speed.

This design felt rather simplistic and uninspiring. It did not have the same 'unique' and 'professional' feeling as the chosen design for the Map View and it was for this reason, the design was not taken further.

Speed Test View Proposal 2

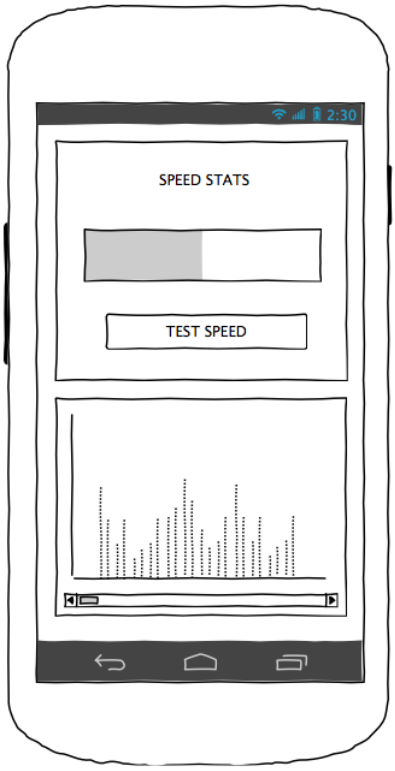


Figure 5.4: **Example 2 of potential Speed Test design**

This second proposal for the Speed Test view used a similar interface to the previous example in the use of a Graph View. This Graph View, again, would show live speed data as described previously.

The main difference in design comes in the use of a progress bar style view to indicate to users how far through the speed test the user is. This not only looks more 'professional', it also fulfils a requirement in the SRS to not leave processes that could last longer than 3 seconds, to do so without informing the user of it's current progress (Requirement P7).

It was felt that this design complemented the overall design of the application so far as it looked modern, intuitive and 'unique'. It was for these reasons that this design was chosen to be developed further and to go into the development phase.

5.3.3 Android's Activity Lifecycle

As these designs will interact with the user, they will be required to extend Android's `Activity` class. This class defines "a single, focused *thing* that the user can do" and will take care of creating a window for the View whilst also allowing the user interface (UI) to be defined using the `setContentView(View)`

function [37].

These Activities, once run, will be managed as an 'activity stack', meaning users can actively move forward and backwards between activities and their UI's. This means that Activities will be able to fall into a number of 'states' which represent where in the activity stack the activity is and how the user can currently interact with it.

The Activity lifecycle is a series of functions[2] that allow developers to programmaticly change what happens to the Activity once it falls into one of these states. These functions are described by the flow chart in Figure 5.5.

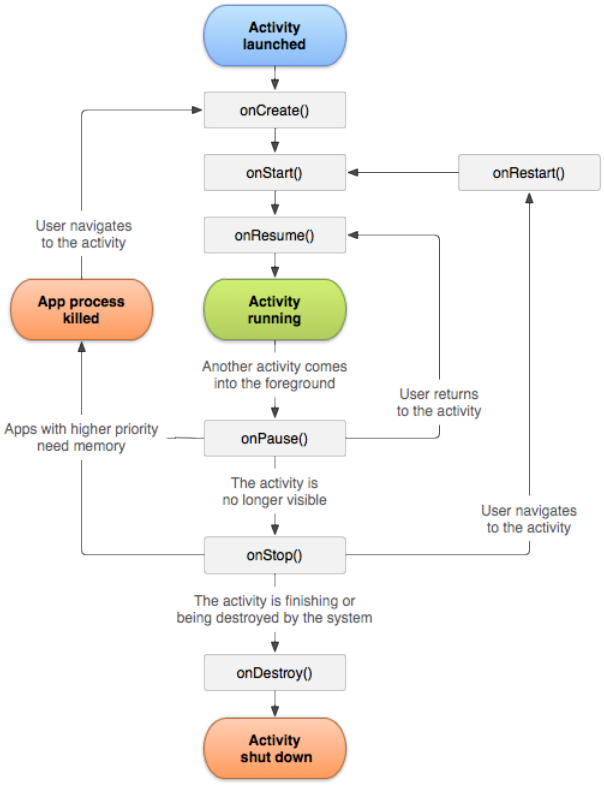


Figure 5.5: **Android Activity Lifecycle**

5.4 Design of additional features

During the implementation of the project, some additional features were added to provided extra functionality and to improve the app as a whole. The design of these features is discussed here but comments regarding their addition to the project and their implementation can be found later in this document.

[2]For more information: `http://bit.ly/ActivityLifecycle`

5.4.1 Base Station Database

The base station database is a MySQL database created with data adapted from the Ofcom provided dataset of mobile base stations within the UK, described previously in Section 3.2.2. A brief description of this dataset and the columns it contains is described below:

Table 5.1: **The Sitefinder Base Station Dataset**

Column Name	Description
Operator	The company in charge of the site
Operator Ref	Company reference to site
Site NGR	National Grid Reference of site location
Height	The height of the antenna in meters
Type	The protocol the antenna serves (e.g. GSM)
Band	The frequency the antenna operates at (MHz)
Antenna Type	The type of Antenna used
Power (dBm)	Power of the Antenna
Max Power (dBm)	Maximum output of the Antenna

From this dataset the following MySQL database was designed. The database uses only fields from the dataset that were deemed relevant and 'informative' to the average user and is therefore significantly smaller.

Table 5.2: **The Base Station MySQL Database Design**

Column Name	Type	Attributes	Extra
Id (Primary Key)	bigint(20)	UNSIGNED	AUTO INCREMENT
Operator	varchar(40)		
Latitude	decimal(30,12)		
Longitude	decimal(30,12)		
Height	decimal(5,2)		
Type	varchar(8)		
Band	int(4)		

The column names directly correspond to the names of the attributes from each record of the dataset that are stored in the MySQL database. For example, the column 'Operator' stores the Base Station's operator. The type of the column name refers to the datatype of the column and the accepted input, should more records be added at a later date.

The number following these types refers to the accepted length of a given field e.g. varchar(8) refers to a string of maximum length 8. The length of each field was carefully considered during the database design, with any potential changes that may occur in the future being carefully considered. It was important to consider these values, as a value that is too small may cause valid data to be seen as erroneous and discarded, whilst having a value too large may cause the

database to be too‘wasteful’ as every record may have fields that are not completely filled with their allotted space.

The only changes between the dataset provided by Ofcom and the database, include the addition of a unique ID for each record and the additional location information extracted from the original data. This process will be described in greater detail further in the report.

6

Implementation

6.1 Introduction

This section describes in detail, the processes involved in the creation of the application. It will cover the tools, libraries and languages used for the different sections of the overall project as well as the various issues encountered during their implementation. Finally, possible improvements will also be discussed. Code fragments are displayed and annotated where possible.

6.2 Adopted Methodology

The development of the application used an agile methodology with the following key points:

- **Working in short iterations**
  The aim was to break the project down into phases. These phases could then be further broken down into two or three week 'sprints'. This provided a renewed focused on what needed to be achieved and gave strong emphasis on time management. In total there were two iterations for both the Map View and the Speed View.
- **Accepting change as normal**
  From previous project experience, it was known that problems during development may cause a project's initial design and requirements to differ significantly during implementation. These 'changes' to design and the project were anticipated from the beginning and addressed as they occurred. An attempt was made to plan before each sprint making it easier to incorporate any changes to the project into the work flow.
- **Deciding as late as possible**
  For most software designs, key details are often unknown at the beginning of a project's life cycle but become clearer as the project progresses. Making decisions as late as possible allows for the most current information

available to be used to form decisions which, theoretically, should lead to better and more informed decisions about the application and it's development.

6.2.1 Storage

The code was backed up on DropBox[1] as it allowed cloud storage and file synchronization between multiple machines. This meant the project was accessible from any machine with an internet connection.

During the development of the project, the code was also periodically backed up using git revision control. Specifically, a private GitHub[2] web repository was used for this purpose. This meant it was easy to keep track of changes made to the local copy of the source code and change back, 'revert', to previous versions of code should the source require it.

6.2.2 Analysis of Methodology

The methodology, in theory, is an excellent example of how teams in professional software houses practice. The simple principles behind it mean that, for a full time project, the work flow is appropriately paced and managed. However, due to other University work constraints, this methodology was occasionally difficult to follow effectively. Despite this, the methodology allowed the project to be completed to a high standard. From this result, the principles may be found to be beneficial for use in any further work.

The use of file synchronization and version control was *invaluable* to the project as it allowed the project to be up-to-date regardless of what machine was editing it. During the project, the development platform was changed from Windows Vista to Apple OSX. Dropbox allowed files from Windows to be automatically downloaded onto the new Apple machine, meaning the project could continue development with minimal disruption. Apple OSX's inbuilt support for `git` also allowed revision control to be remarkably easy. This suggests the use of these tools was appropriate and that any further work or projects should also consider using similar utilities.

6.3 Development Environment

The development environment that the system was developed on was initially a Windows Vista machine. This was later switched to an Apple Mac machine. The specifications of the two machines are detailed below:

[1]Dropbox: `http://dropbox.com`
[2]GitHub: `http://github.com/ed-george`

Table 6.1: **Specifications of Development Machines**

	Machine 1	Machine 2
OS	**Windows Vista**	**OSX 10.7.5**
Hard Drive	**300GB**	**750GB**
RAM	**4GB**	**8 GB**
Graphics	**512MB**	**1024MB**
Processor	**2 GHz Intel Core 2 Duo**	**2.4 GHz Intel Core i7**
Android API	**15**	**15**
Android SDK	**18**	**21**
IDE	**Eclipse: Indigo**	**Eclipse: Indigo**

6.4 Platform

6.4.1 Choice of Mobile Platform

In recent years,'smart phones' have become ever more popular across the globe [38]. Phones can now provide additional functions such as e-mail, GPS navigation, internet browsing and media capture in addition to their purpose as a communication device.

These phones are built on differing computing platforms dependant on their manufacturer and serve to provide the interface between the phone's hardware and user's input. There are multiple platforms that are in use globally including Android, Apple's iOS and Blackberry OS by Research In Motion (RIM). Each platform has a different look and feel for users, as well as a different 'back-end' and software development kits (SDK) for mobile platform developers.

These platforms allow developers to create and distribute applications known as 'apps' to smart phone users through dedicated app stores. The two most popular app stores are Google's Play Store for Android and Apple's App Store for iOS. During early development it was these two platforms that were considered for the project's development.

Android was eventually chosen as it is currently the most popular mobile software platform in use around the world with around 1 million new devices being activated every 24 hours [7]. It was also chosen as it's SDK and other tools are free to use and available across multiple platforms including Windows, Linux and Mac OSX. This is not the case for iOS as it requires development to be based on an Apple Mac machine as well as a paid subscription to Apple's Developer Program before any product can be released to their App Store[3] . Initially, the development environment used was incompatible with iOS development, meaning Android development seemed more logical.

[3] This is $99 p/year currently, whilst only $25 is required to release on Google Play [8]

6.4.2 Choice of Programming Languages

The Java programming language was used for the majority of the project due to the Android SDK using Java as the default language for application development. The Android platform does allow applications to be written in other programming languages such as C/C++[4], scripting languages such as Python, Perl and Lua[5] or HTML 5 and other web languages[6] but Java was selected due to its excellent documentation within Android, ease of use and extensive libraries available to developers - including third party libraries. Java also fitted the methodology adopted as development could be rapid in comparison to C/C++ development due to Java's 'garbage collector' approach to memory allocation and deallocation, a feature not present in C/C++.

For the Mobile base station database, PHP was selected as the server side language. This was due to PHP's excellent library functions that support MySQL, as this was used as the Database management system. It was also chosen as PHP's syntax is similar to that of languages such as C/C++ and Java, making it easy to learn.

6.4.3 Choice of IDE

An integrated development environment (IDE), is a piece of software used for software development. There are many professional IDEs available that offer a variety of functions and support for Java applications. However, the Android SDK provides a 'plug-in' for the popular IDE Eclipse. This 'plug-in' integrates Android and its tools into the development environment including debugging software and full project support.

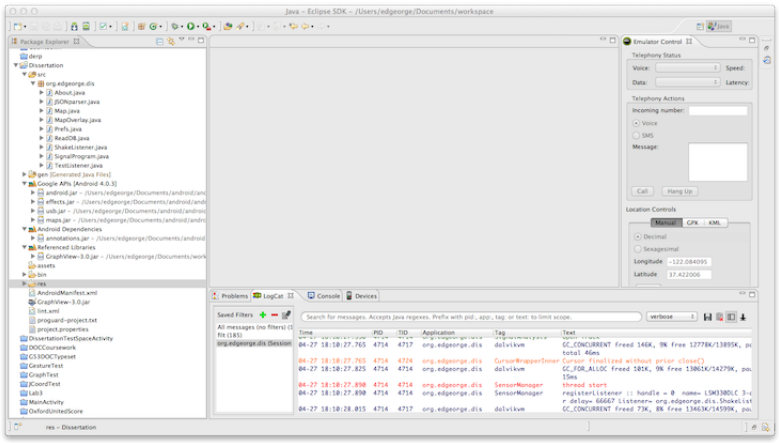


Figure 6.1: **The Eclipse IDE with Android plug-in**

[4] Using Android's Native Development Kit
[5] Using the Android Scripting Environment
[6] Through the use of Adobe's PhoneGap `http://phonegap.com`

Eclipse allows projects to be built instantly either through the Android Emulator, a program designed to emulate the behaviour of an Android device, or through a USB connection to a physical Android device. This makes debugging quick and thorough, as well as allowing applications to be tested on multiple device types.

Other Java IDEs such as IntelliJ, NetBeans and JCreator do not offer these features natively and would require command line support to perform the same tasks meaning Eclipse was chosen due to it's simplicity of use.

6.4.4 Libraries Used

The application uses some native Android libraries and other Java libraries to provide extra functionality. These are detailed below.

Google/Android Libraries

The application makes use of many default Android libraries. This is to support key features and concepts of the application and is standard practise when developing for the Android platform. These libraries range in functionality and can be used to provide access to features of a device such as the GPS or camera, as well as to access phone information such as contact data or used to create layouts and track user input. These libraries are identifiable through their `android.*` package names at the beginning of the Java files used to create the application.

In addition to these Android libraries, further Google libraries are used. The `com.google.android.maps` library provides support for Google Maps integration within Android applications [39] and was used for the Map View interface. This library was chosen as Google Maps provides good quality up-to-date maps and a well documented API for development on the Android platform. The library also had support for 'Map Overlays[7]' meaning points of interest could be plotted on the map and shown using graphics [40].

External Libraries

The Graph View that is used within both the two key interfaces is generated partially using a third-party library. 'GraphView' is a third party Android library that can be used to produce Line Graphs or Bar Graphs easily [41]. It is created by Jonas Gehring and released under the GNU Lesser General Public License (LGPL).

[7] `com.google.android.maps.Overlay`

The latest version of the GraphView library[8] allows data to be added to a graph in 'real time'. This feature was desired in the designs of the interfaces as described in Section 5.3, and for this reason the external library was used. Alternatives to GraphView such as ChartDroid[9] or AChartEngine[10] do exist but were deemed too complex for the task at hand due to the large variety of customization and graph types supported.

For the communication between the Web based Mobile Mast database and the application, the `org.apache.http` and `org.json` libraries were used. The `org.apache.http` library is used to communicate from the client (device) to the server via the Hyper Text Transfer Protocol (HTTP) protocol to collect mast information [42].

The `org.json` library was used to extract the information received from the server and process it to provide an output that could be given to the user [43].

The method of how this was completed is described further later in this document.

6.4.5 Other Tools Used

Other tools and utilities were used during the implementation of the project and these are detailed below:

- **phpMyAdmin**
  phpMyAdmin is a free tool to manage MySQL databases via a web browser [44]. It can be used to complete numerous database functions such as creating, modifying or deleting databases and tables, as well as executing SQL queries.

- **Android Tools**
  The Android SDK provides numerous tools to aid developers in creating applications. These include the Android Virtual Device (AVD), an Android emulator configuration that can be used to model an actual Android device by defining specific hardware and software options to be emulated and run on the Android Emulator [45].
  The Android emulator, is a program that represents a virtual mobile device, both aesthetically and functionally, on a computer. This enables applications to be developed and tested without the use of a physical device [46].

[8] `com.jjoe64.graphview`
[9] `https://code.google.com/p/chartdroid/`
[10] `https://code.google.com/p/achartengine/`

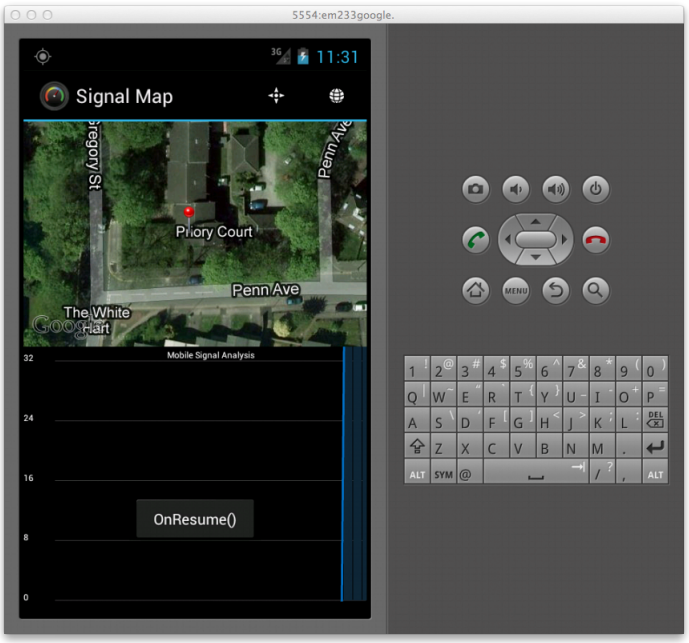


Figure 6.2: **The Android Emulator running a development version of the Map View interface**

6.5 Implementing the Map View Interface

The Map View interface, implemented in `Map.java`, is one of the two key interfaces used in the application and is based on the final design as proposed in Section 5.3. The implementation of the design is comprised of a vertical Linear Layout containing a Frame View and a further Linear Layout for the Map and Graph respectively. The implementation was split into these two respective sub-views.

The class extends Android's Activity class, as their will be interaction with the user. The Activity class takes care of creating a window for this interaction to take place and implements the 'activity lifecycle', a design methodology and set of functions that handle the state of the activity.

6.5.1 Live Graph View

Part of the interface created by the Map Activity within the application provides users with a live graph mapping their current signal strength.

This view makes use of the imported third-party libraries provided by `com.jjoe64.graphview.*` which allows objects of type `GraphView`, `GraphViewSeries`, `LineGraphView` to be created and used within the application. These objects are used to create and initialise a new graph and reference the data it is required to plot.

The graph is initialised within the Map Activity `onCreate()` method, the default function called by Android once an Activity is viewed[11], as follows:

```
@Override
  public void onCreate(Bundle savedInstanceState)
  {

    //...More Code Here

    //Create new GraphData series with single data point (1,0)
    graphViewData = new GraphViewSeries(new GraphViewData[] {
        new GraphViewData(1, 0.0d)});

    //Create new Line Graph titled "Current Signal Strength"
    graphView = new LineGraphView(this, "Current Signal Strength");

    //Draw blue background underneath line graph
    ((LineGraphView) graphView).setDrawBackground(true);

    //Add the associated data series to the graph
    graphView.addSeries(graphViewData);

    //Locate the graph's location in layout from layout file
    LinearLayout layout = (LinearLayout) findViewById(R.id.graph1);

    //Add graph specific params
    graphView.setManualYAxisBounds(32.0, 0.0);
    graphView.setViewPort(0,50);
    graphView.setHorizontalLabels(new String[] {""});
    graphView.setScrollable(true);

    // Add the Graph to the Layout
    layout.addView(graphView);

    //...More Code Here

  }
```

This code is used to create the graphs and initialise its dataset. To periodically update the graph with signal information, the Map class uses a `TestListener` object. `TestListener.java` is an implementation of the base class `PhoneStateListener`[12] which provides the abstract function `onSignalStrengthsChanged(SignalStrength)` that is invoked when network signal strengths changes [48]. This allows the values of the signal strength to be stored in variables and then accessed through the created 'getter' functions `getGSMSig()` or `getCDMASig()`.

This live signal strength was initially measured in decibels (dBm) but was changed to match absolute signal units (ASU) values set by the European

[11] See Figure 5.5

[12] `android.telephony.PhoneStateListener`

Telecommunications Standards Institute (ETSI) in the TS 27.007 8.5 standard. This was due to decibel signal strength being measured in negative numbers and looking counter intuitive on a graph due to the scale being below the origin. The valid values set out by ETSI are 0-31 for valid signal and 99 for error values [49] and can be converted relatively simply to dBm, denoted as P in the following equations:

$$P = (ETSI * 2) - 133$$
$$ETSI = \frac{P + 133}{2}$$

The `SignalStrength`[13] object passed into the `onSignalStrengthsChanged()` method has functions to get the current GSM, `getGsmSignalStrength()`, and CDMA strengths, `getCdmaDbm()`[50]. It is through these methods, the `TestListener` gets access and stores the current signal data. It should be noted that the `getGsmSignalStrength()` method gives the device's current GSM strength in the ETSI standard as previously described, but `getCdmaDbm()` is returned in decibels (dBm).

To periodically update the graph, a `Runnable` object is created in the `onResume()` method[14] of the Map Activity in `Map.java`. The `Runnable` type is used as it allows a command to be executed periodically through a `Handler` [51].

```
@Override
  public void onResume(){
    super.onResume();
    //Restart listeners
    tm.listen(tl,PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
    man.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10000,
        0, this);
    mSensorManager.registerListener(mSensorListener,
        mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
        SensorManager.SENSOR_DELAY_UI);
    //Use handler to handle a runnable and
    signalRefreshTimer = new Runnable() {
      public void run() {
        //increase the x value (time)
        lastXValue += 1d;
        //Append new latest Signal Data to graph
        graphViewData.appendData(new GraphViewData(lastXValue, getSIG
            ()), true);
        graphView.setHorizontalLabels(empty);
        //Update
        mHandler.postDelayed(this, duration);
      }
    };
    mHandler.postDelayed(signalRefreshTimer, duration);
  }
```

[13] `android.telephony.SignalStrength`
[14] See Figure 5.5

The previous code uses a new instance of the `Runnable` class and it's abstract method `run()` that is called and executed periodically ever `duration` seconds[15] by `mHandler.postDelayed()`. Within the abstract `run()` method, the Graph View's data is updated through `getSig()` that is defined below:

```
1  private double getSIG() {
2    //Uses TestListener tl
3    if(tl.isGSMPhone()){
4      //Return GSM
5      return (double) tl.getGSMSig();
6    }
7    //Return CDMA
8    return (double) tl.getCDMASig();
9  }
```

Further to the requirements set out in the SRS, a Shake Listener was implemented. This class is designed to actively monitor if the device is shaked, and refresh the graph with no data should a 'shake' be detected.

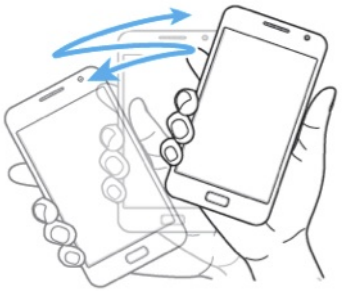


Figure 6.3: **An example of a 'shake' [52]**

This was completed simply through accessing the device's inbuilt accelerometers, a set of sensors used to measure the acceleration force applied to a device on all three physical axes (x, y, and z) [53]. This is implemented in `ShakeListener.java`. This class implements the `SensorEventListener` base class to access the changes in the accelerometers through the `onSensorChanged(SensorEvent)` method [54].

[15] Currently defined as every 0.5s

Using this class, a listener can be created in the `onCreate()` method within `Map.java` to initialise the listener as soon as the Activity begin and define what should happen `onShake()`.

```
@Override
  public void onCreate(Bundle savedInstanceState)
  {
    //...More Code Here

    //Set up the manager of listener to sensors
    mSensorManager = (SensorManager) getSystemService(Context.
        SENSOR_SERVICE);
    //Create new listener for 'Shakes'
    mSensorListener = new ShakeListener();
    //Handle Shake
    mSensorListener.setOnShakeListener(new ShakeListener.
        OnShakeListener() {

      //On shake, reset graph data
      public void onShake() {
        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
          public void run() {
            //Reset Data Series
            graphViewData.resetData(new GraphViewData[] {
                new GraphViewData(1, getSIG())});
            //Post message to screen telling user of Data Reset
            Toast.makeText(Map.this, "Data Reset", Toast.LENGTH_LONG)
                .show();
          }
        }, 2000);
      }});

  }
```

6.5.2 Live Map View

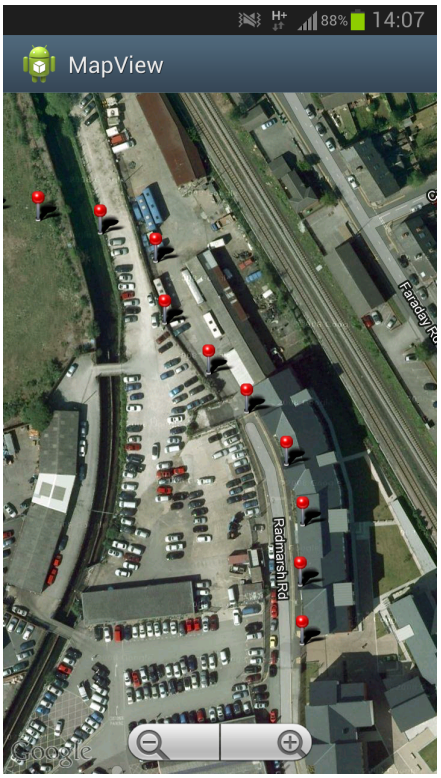


Figure 6.4: **An early prototype of the Map View**

The Map within the Map activity is created using the external Google Maps third party library. This map displays the current user location and overlays as described in the UI design specification.

To create the map, a request for a Google Maps API v1 key[16] was used to allow the application access to the Google Maps API and subsequently allow maps to be embedded within the Activity using the following code in the XML layout file for the Map Activity.

```
<com.google.android.maps.MapView
        android:id="@+id/map"
    android:layout_width="fill_parent"
   android:layout_height="match_parent"
        android:enabled="true"
       android:clickable="true"
    android:apiKey="api-key-here"/>
```

The map view and its attributes, such as the default zoom height and overlays, are defined within the Map Activity's `onCreate()` method. The main map interaction comes from the `Map` class implementing `LocationListener` and the

[16] The project began before the release of Google Maps API v2 - released in December 2012

`onLocationChanged()` method that handles what to do when the mobile device is registered at a new current location. This method begins by checking if a new overlay should be placed at the new location, by checking the distance between the current location and the last known location and comparing this against a threshold. Should an overlay be required, the updated location is created and placed on the map along with the overlay and the appropriate signal data for the location.

This is demonstrated in the code below:

```
public void onLocationChanged(Location location) {
    //Called when current GPS location has changed
    if (shouldAddOverlay(location)){
      //create new location
      GeoPoint point = new GeoPoint((int) (location.getLatitude() * 1
          E6), (int) (location.getLongitude() * 1E6));
      int gsm = tl.getGSMSig(); //get phone signal data
      int cdma = tl.getCDMASig();
      //move map to current location
      mapView.getController().animateTo(point);
      if(tl.isGSMPhone()){ //check phone type
        //change icon depending on signal strength
        locationOverlay.setIconOverlay(gsm);
      }else{
        locationOverlay.setIconOverlay(cdma);
      }
      //Create new overlay and information it displays in it's dialog
      OverlayItem newOver = new OverlayItem(point,"Point: " + ++
          graphdata_points , makeString(gsm, cdma, location));
      //add this overlay into overlay list
      locationOverlay.addOverlay(newOver);
      //add the overlays to map
      mapOverlays.add(locationOverlay);
    }
    //update last current location known
    lastKnownLat = location.getLatitude();
    lastKnownLon = location.getLongitude();
  }
```

The `OverlayItem` class' constructor allows the customisation of the dialogue viewed once an Overlay has been 'clicked' by the user. This dialogue can be viewed in Figure 6.5.

Further to the Overlays, the map was also added with an options menu containing options to revert the map to Road view, to revert to Satellite view and to add base station overlays to the map. These options were defined in `/res/menu/mapmenu.xml` and describe the three basic options in terms of layout. They were implemented using the `onCreateOptionsMenu()` and `onOptionsItemSelected()` functions that were both extended from the Activity base class.

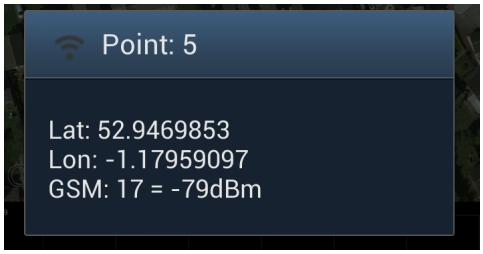


Figure 6.5: **An example of the dialogue box produced by an Overlay**

6.5.3 Problems Encountered

During the implementation of the Graph view, there were a number of issues that occurred causing issues with the graphs performance.

The main issue experienced was the graph not updating correctly on a physical mobile device. The graph would remain stationary until a user interacted with it, causing the graph to update and display the correct data. This was due to the graph performing it's actions on Android's UI Thread, the main thread of execution for an application. This thread is where application components such as the Activities and Services are created and anything that causes the UI to be updated is required to be called on this Thread. Having a Map View that updates on the same Thread, meant the Graph was often not executed by the thread causing it to appear frozen, despite the background code working normally.

To rectify this problem, the Android OS library provides the `AsyncTask` class that allows the current task to perform background operations and publish results onto Android's UI thread without having to manipulate other threads or handlers. A subclass for the map view that extends `AsyncTask` should have been created to perform the updates to the User Interface.z

Unfortunately, due to time constraints, this issue has not been rectified currently and remains as a small issue for the application.

6.6 Implementing the Speed Test View

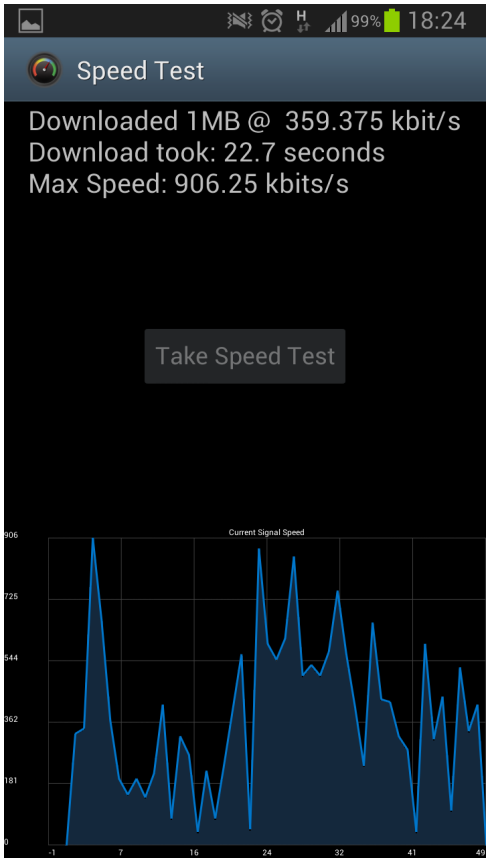


Figure 6.6: **The Speed Test Activity after running a test**

The Speed Test activity, created in `SpeedTest.java`, is the implementation of the Speed Test design as specified in Section 5.3. This activity provides the user information about their current 3G signal speed and is completed using the following method.

The Activity uses an executable `Runnable` to download a 'dummy file' of a known size[17] from some private webspace. During this download, the average speed, total download time and maximum download speed are recorded and returned to the user after completion as shown in Figure 6.6. During download, a Graph View is updated with the current download speed.

The `Runnable`, named `downloadRun()`, downloads the dummy file byte by byte and updates the UI to its progress using a `Message` passed to a `Handler`.

[17] File size is 1MB which is 1024KB and equal to 1048576 bytes in total

```
1  //Read single byte at a time from download stream
2          while((currentByte = stream.read()) != -1){
3            //Increase overall bytes received
4            bytesRecieved++;
5            //Number of bytes in current threshold
6            bytesCurrentThreshold++;
7            //If time to update lasted longer than minimum threshold
8            if(updateTime >= DL_TIME_THRESHOLD){
9              //work out current progress
10             int progress= (int) ((bytesRecieved/(double)
                   EXPECTED_SIZE) * 100);
11             //Send message to handler telling UI to be updated as
                   more bytes have been recieved - send current speed
                   info
12             Message msg = Message.obtain(threadHandler,
                   UPDATE_OCCURED_MESSAGE, getSpeedInfo(updateTime,
                   bytesCurrentThreshold));
13             //Also send progress
14             msg.arg1 = progress;
15             //Send message to handler
16             threadHandler.sendMessage(msg);
17             //reset update start
18             //reset bytes in threshold
19             updateStart=System.currentTimeMillis();
20             bytesCurrentThreshold = 0;
21           }
22           //Overall time taken to update
23           updateTime = System.currentTimeMillis() - updateStart;
24         }
25         //Total time to download
26         totalDownloadTime = (System.currentTimeMillis() - start);
27         //Message handler telling of completion
28         Message msg = Message.obtain(threadHandler, COMPLETE_MESSAGE,
               getSpeedInfo(totalDownloadTime, bytesRecieved));
29         Toast.makeText(SpeedTest.this, "Complete", Toast.LENGTH_SHORT
               ).show();
30         threadHandler.sendMessage(msg);
31       }
```

As `threadHandler` receives the messages from the `Runnable`, it is used to 'alter the UI' both during download and once download is complete. Altering the UI includes providing up-to-date information on the download progress through updating the progress bar and progress text with the current percentage of progress completed. It also involves updating the Graph View with the current download speed in Kilobits per second.

This `Handler`'s functionality is briefly described in the code sample below:

```
//More Code...
      case UPDATE_OCCURED_MESSAGE:
        //get current speed info passed from runnable
        final CurrentSpeedInfo current = (CurrentSpeedInfo) msg.obj;
        //if the current speed is faster than the max overall speed,
            the new speed is the max
        if(current.kb > maxSpeed){
          maxSpeed = current.kb;
        }
        last_x += 1d; //update graph x value
        //Append new latest Signal Data to graph using passed object
        graphViewData.appendData(new GraphViewData(last_x, current.kb
            ), true);
        //Update text view with current % download
        tv.setText("Downloading " + Integer.toString((int) msg.arg1)
            +"%");
        //update progress bar
        pb.setProgress(msg.arg1);
        break; //More Code...
```

A `Runnable` and `Handler` were used as this allowed background work from the `Runnable` to update the UI. Alternatively, an `AsyncTask` could have been used but this would have been over complicated and produced a similar result. It was for this reason that the implementation of the Speed Test view was completed as such.

6.6.1 Problems Encountered

During the implementation of the Speed Test, it was initially unclear how to provide a dummy file of a set size. The was solved using the `dd` command that is available for '*nix' based machines, including the Apple OSX machine used as the development environment for the majority of the project. This command allows files to be copied and created with a certain number of bytes [55] as well as specify the input and output streams.

Therefore, the following function was used to create a file of exactly 1MB named 'dummy.txt':

```
dd if=/dev/zero of=dummy.txt bs=1048576 count=1
```

A further problem encountered, was the progress of the download being stuck at 99%. This occurred due to a `Toast` message being called on the Worker thread within `downloadJSON()`. This `Toast` message was attempting to draw on the UI thread from the Worker thread which is strictly prohibited within Android. This was rectified by moving the `Toast` message to the `Handler` more specifically the

message associated with completion of the download.

During testing, it was observed that should the user return to any other Activity on the activity stack or focus from the Speed Test activity was lost, the speed test would continue and the download of the 1MB would remain running in the background. This was not desirable as the user's actions would suggest they wished to 'cancel' the running of the speed test and stop the download of the file used to measure the signal speed. This problem was rectified by creating a public boolean value named `shouldContinue` within the main loop of the Thread that was set to `true` by default and switched to `false` once the activity' `onPause()` or `onDestroy()` methods were called. Using the logical AND operator, (`&&`, this allowed the loop to fail once the boolean value became false, subsequently stopping the download of the file.

```
while((currentByte = stream.read()) != -1 && shouldContinue){
          //Increase overall bytes received
          bytesRecieved++;
          //Number of bytes in current threshold
          bytesCurrentThreshold++;
  //etc...
```

Other problems encountered included finding out the conversion rates between Kilobit and Megabit. To complete this, an online tool[18] was used to find the ratio between the two units.

6.7 Implementing the Mobile Base Station Database

The Mobile Mast/Base Station database is an adaptation of Ofcom's Sitefinder dataset[19] and it's data. Initially, as described in Table 5.1, the dataset provided over 140,000 records of mobile antennas and their location. Each record's location was provided as a 'National Grid Reference' (NGR) an Ordinance Survey standard that uses 100km squares identifiable through a letter pair, followed by digits to identify a sub-square within the larger grid [56]. These sub-square's origins are located in the south west corner of the 100km grids, similar to the first quadrant in Cartesian geometry[20].

A 6-digit reference following the letter-pair identifies a 100m grid sub-square, a 8-digit references a 10m grid sub-square, and 10-digits identifies single 1m sub-squares. These reference numbers represent the x-coordinates known as 'Eastings' and y-coordinates known as 'Northings' in kilometres.

Used as an example of one of the locations provided, **SZ4356381685** represents a 1m box due to it's 10 digit reference, with it's south west origin 43.563km

[18] Found at: http://www.matisse.net/bitcalc/
[19] This can be found here: http://stakeholders.ofcom.org.uk/sitefinder/sitefinder-dataset/
[20] For more information visit: http://www.ordnancesurvey.co.uk/oswebsite/gi/nationalgrid/nghelp2.html

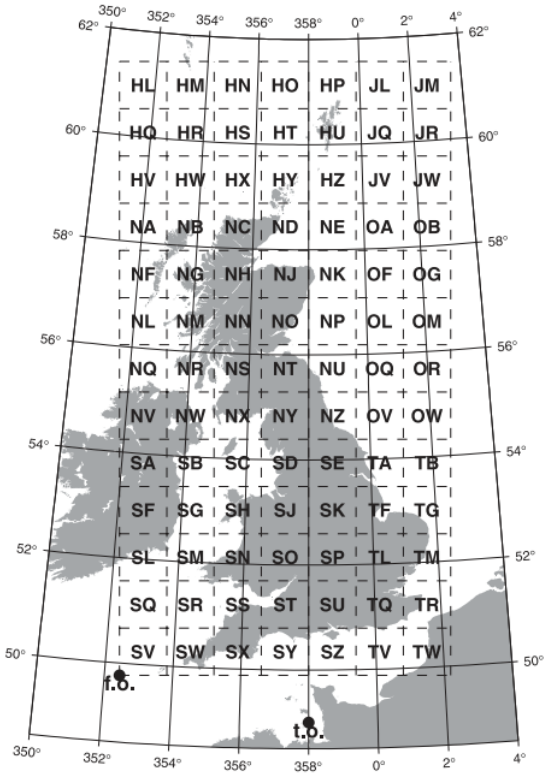


Figure 6.7: **The NGR layout within Great Britain**

across and 81.685m up within the SZ square.

This location protocol is not used outside Great Britain and, therefore, is not supported by Google Maps and the Google Maps API. The required input was in the form of Latitude and Longitude, a globally used geographical coordinate system. This posed as a problem as all 140,000 records would require conversion to the Lat/Lon system and there was no clear way of converting the Microsoft Excel file dataset's columns to the desired format.

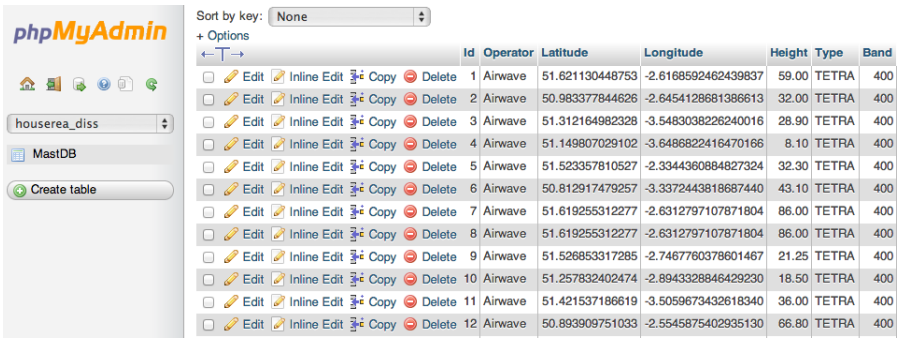
To convert the dataset's locations to the desired format, the data was first exported into a Comma Separated Value (CSV) file. This allowed better access to the data than the Excel file as CSV files are syntactically very easy to produce and read. Rows from the dataset were represented as single lines within the CSV (i.e. using the newline as a 'delimiter'), whilst the multiple fields were separated using commas.

After some research, the JCoord library, an external library created by Jonathan Scott, was discovered to contain functions that could convert NGR references into Lat/Lon coordinates [57]. A Java class was created along with the JCoord library[21] that parsed the CSV file, extracted the information deemed relevant to the application, added the converted latitude and longitude values from the NGR reference and printed the result into a new CSV file on the standard output.

[21] `uk.me.jstott.jcoord`

```
    //More Code Previously...
    while ((line = bufferedReader.readLine()) != null) {
      //Split the current line into sub strings by a comma
      //I.e. get columns of dataset
      String[] tokens = line.split(",");
      //3rd Column is the NGR
      //Create new OSRef object with Location
      OSRef x = new OSRef(tokens[2]);
      //Convert to LatLon object, then retrieve lat and lon
      String lat = Double.toString(x.toLatLng().getLat());
      String lon = Double.toString(x.toLatLng().getLng());
      //Print line to StdOut
      System.out.println(tokens[0] + "," + lat + "," + lon + "," +
          tokens[3] + "," + tokens[4] + "," + tokens[5]);
    }
    //Close Readers
    bufferedReader.close();
    fileReader.close();
  }
```

This code produced a new CSV file that contained the fields required for the MySQL database design as described in Figure 5.2. Through the use of some personal private web space, the new CSV file was able to be imported by phpMyAdmin's import tools and allowed the creation of the database.



	Id	Operator	Latitude	Longitude	Height	Type	Band
Edit Inline Edit Copy Delete	1	Airwave	51.621130448753	-2.6168592462439837	59.00	TETRA	400
Edit Inline Edit Copy Delete	2	Airwave	50.983377844626	-2.6454128681386613	32.00	TETRA	400
Edit Inline Edit Copy Delete	3	Airwave	51.312164982328	-3.5483038226240016	28.90	TETRA	400
Edit Inline Edit Copy Delete	4	Airwave	51.149807029102	-3.6486822416470166	8.10	TETRA	400
Edit Inline Edit Copy Delete	5	Airwave	51.523357810527	-2.3344360884827324	32.30	TETRA	400
Edit Inline Edit Copy Delete	6	Airwave	50.812917479257	-3.3372443818687440	43.10	TETRA	400
Edit Inline Edit Copy Delete	7	Airwave	51.619255312277	-2.6312797107871804	86.00	TETRA	400
Edit Inline Edit Copy Delete	8	Airwave	51.619255312277	-2.6312797107871804	86.00	TETRA	400
Edit Inline Edit Copy Delete	9	Airwave	51.526853317285	-2.7467760378601467	21.25	TETRA	400
Edit Inline Edit Copy Delete	10	Airwave	51.257832402474	-2.8943328846429230	18.50	TETRA	400
Edit Inline Edit Copy Delete	11	Airwave	51.421537186619	-3.5059673432618340	36.00	TETRA	400
Edit Inline Edit Copy Delete	12	Airwave	50.893909751033	-2.5545875402935130	66.80	TETRA	400

Figure 6.8: **The implemented MySQL database**

This led to the main issue at hand, '*How best to communicate from the web based database to the client on a mobile device?*'.

To solve this problem, PHP, JavaScript Object Notation (JSON) as well as Apache's HTTP library were used in the following topology:

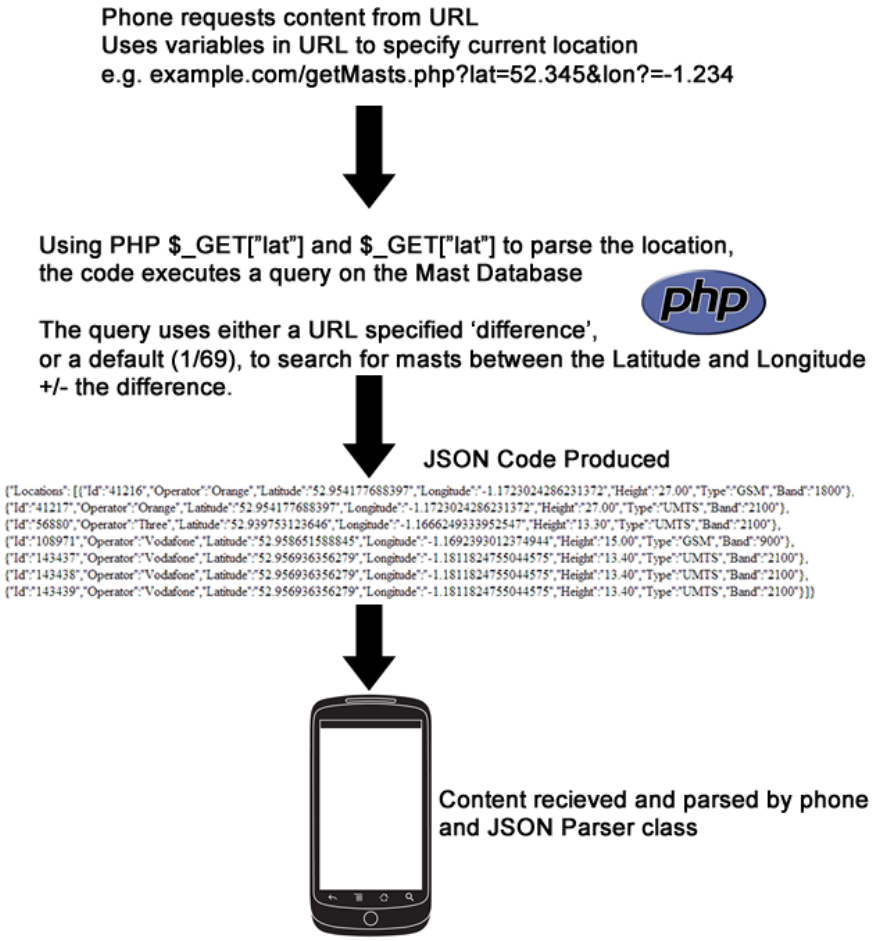


Figure 6.9: **The method used to communicate between the client and server**

The main PHP code used to complete the server side functions to communicate to the database is summarised below.

```
1  //Store values from URL
2  $lat = $_GET["lat"];
3  $lon = $_GET["lon"];
4  $diff = $_GET["dif"];
5  $debug = $_GET["debug"];
6  $latS = $lat - $diff;
7  $latL = $lat + $diff;
8  $lonS = $lon - $diff;
9  $lonL = $lon + $diff;
10 //Find appropriate records using MySQL Query
11 $sth = mysql_query("SELECT * FROM MastDB WHERE Latitude BETWEEN
       $latS AND $latL AND Longitude BETWEEN $lonS AND $lonL");
12 $rows = array();
13 while($r = mysql_fetch_assoc($sth)) {
```

```
14      $rows[] = $r; //Add results to $rows[]
15    }
16    echo "{\"Locations\": "; //Correct JSON Formatting
17    print json_encode($rows); //print json results
18    echo "}";
```

The code used a MySQL query along with a set 'difference' to find mobile base stations within a set area of coordinates. However, this may have been achieved better by first setting up a query for the value produced when taking the absolute value of the difference between the database location values and the input value of the user's current location. This query could then be sorted by increasing value and limited to a number of results. This would allow the nearest n masts to be found where n is the limit set within the statement.

JSON was used as PHP 5.2 has built in functions to turn the result of a MySQL query into JSON notation [58]. Other mark up languages such as XML could have been used as well as other server side languages such as Perl or ASP.NET.

XML was not used due to any XML output generated being larger and slightly more complex to break into its individual data elements. However, Android does support the `XmlPullParser` class provided by the `org.xmlpull` library[22]. This library could have been used to parse any formatted XML result and should be considered for any future work.

Perl and other server side scripting languages were not used due to the constraints of the private webspace used to host the PHP script and MySQL database. This private webspace did not support scripting languages other than PHP 5.3+, this left no alternative option but to use PHP to access the database.

The JSON data, created through the PHP script, can then be interpreted easily by the mobile device by first capturing the JSON content from the webpage, through Apache's HTTP library, then parsing the data using a custom built parser class.

This approach is used in the `Map.java` activity to display the locations of Mobile Masts that are local to the user on the Google map view. A public class called `DownloadJSON` that extends the `AsyncTask` base class was created to perform the background access to the database and post results to Android's UI thread. The use of `AsyncTask` allowed the Map Activity to perform asynchronous work on the user interface. The extension of the base class `AsyncTask` allows `DownloadJSON` to implement the three functions `onPreExecute()`, `doInBackground()` and `onPostExecute()`. These three functions can be used to define how the user interface should behave before, during and after the downloading and interpretation of the JSON data as shown in Figure 6.10.

[22] For more info: http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html

Figure 6.10: `DownloadJSON`**'s use of the AsyncTask functions**

The `DownloadJSON` class first initiates and produces a 'progress spinner' in the foreground of the application via the `onPreExecute()` method to alert the user that the data from the PHP generated JSON is being downloaded.

This 'spinner' remains during the call to `doInBackground()`. This function downloads the data from the website using the `JSONparser` class and parses the JSON results, creating new map overlays for the individual base stations found and places them on the map before completing and calling the `onPostExecute()` function. This function cancels the progress dialogue box, displays a `Toast` message to the screen to alert the user the process has been completed and allows the user to begin further interaction with the UI.

The JSON is downloaded using this method within the `JSONparser` class:

```
1  public JSONObject downloadJSON(String url) {
2      try {
3        DefaultHttpClient client = new DefaultHttpClient();
4  //GET retrieves whatever information (in the form of an entity) is
       identified by the Request-URI
5        HttpGet httpGet = new HttpGet(url);
6  //Executes a request using the default HTTP client and processes the
       response using the given response handler HTTP GET
7        HttpResponse response = client.execute(httpGet);
8  //Return majority of HTTP request including some of header and full
       body
9        HttpEntity entity = response.getEntity();
10       is = entity.getContent(); //Retrieve Body
11     } catch (Exception e){
12       e.printStackTrace();
13     }
14     try {
15       //Read input stream from HTTP GET body
16       BufferedReader reader = new BufferedReader(new
           InputStreamReader(is, "utf-8"), 8);
17       StringBuilder sb = new StringBuilder();
18       String line = null;
19       //Read in body line by line
20       while ((line = reader.readLine()) != null) {
```

```
21          //append to string builder
22          sb.append(line + "\n");
23        }
24        is.close(); //close stream
25        json = sb.toString(); //stringbuilder contains the JSON
26      } catch (Exception e) {
27        e.printStackTrace();
28      }
29      try {
30        //Return string containing json as JSON object
31        jObject = new JSONObject(json);
32      } catch (JSONException e) {
33        e.printStackTrace();
34      }
35      return jObject;
36    }
```

6.8 Additional Implementation

Further to the three key areas of implementation described, the application was given further smaller features to add to the overall application in addition to the software requirement specification. This were added in the hope to make the application better for users and to provide different functionality.

6.8.1 Home Screen

A simple home screen interface was designed that allowed users to move between the key interfaces by tapping on buttons. This was created to be basic as it was hoped that this would make the application easy to use and intuitive, meaning users should instinctively be able to use the interface.

The Home Screen class (`SignalProgram.java`) extended the `OnClickListener` base class to provide support for when each button of the interface was clicked. On clicking these buttons, the activity they represent is opened using an `Intent` and `startActivity()` method.

For example:

```
1 Intent i = new Intent(this, About.class);
2 startActivity(i);
```

This method adds the home screen activity to the activity stack, meaning users can navigate backwards within an activity to go back to the home screen and the options it provides.

A landscape view was also designed and implemented with a similar interface to that of the standard portrait view.

6.8.2 About Dialog

A short About screen was produced to give users information about the application should they be interested. This was created by defining the theme of the Activity to be `"@android:style/Theme.Dialog"` within the Manifest file. This creates an Activity to appear like a dialog box in the centre of the screen. The layout of the About dialog just displays a pre-defined string that contains information about the program.

6.8.3 Base Station Information

Further to the overlays available within the Map View, a new Activity was created that made use of the mobile base station database previously described. This Activity, called 'Access Mast Data', was designed to show information about local mobile base stations in the form of text, rather than on a map to show their location. This text is the same as that featured in the Map View when a base station overlay is tapped to bring up the relevant information.

This text view was the result of a prototype for collecting information from the JSON webpage. It uses the same `ASyncTask` as described previously in Section 6.7 to access information from the MySQL database before iterating through the resulting array and concatenating results together to form a large list of nearby base stations. This Activity uses the same code as previously described for the Map View interface with very few modifications

The Activity actively monitors the phone's location through GPS and this is used to find the current location that is used to search for local base stations. Through monitoring changes of location, it is also possible to stop the user from making multiple requests to view local data if they have not moved a significant distance from their last known location. A message is displayed should a user attempt this and was implemented as it stops unnecessary requests being made.

The UI used a single button to request information from the database and a text view to display the results. Once again, as seen in the Map View, a loading 'spinner' was used to alert users that the download of information from the database was taking place. This was added as the process works in the background of the application and would otherwise not display anything to let users know of its process.

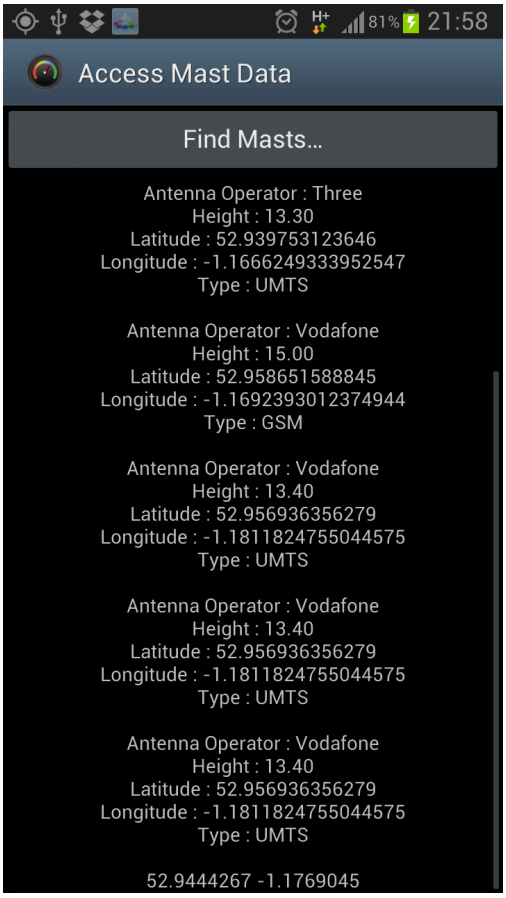


Figure 6.11: **The UI of the Mast Data Activity**

6.9 Implementation Summary

The application is a mobile application that can be categorised into three key parts - a Speed Test utility, a Map View utility and the underlying mobile mast database hosted on private webspace. The two utilities are Android based activities, programmed in Java that provide the user with information about the properties of the mobile signal their mobile device is receiving in real time. These utilities make use of Android libraries as well as external libraries such as Google Maps and Apache HTTP to provide users with this live detailed information and update the UI. The database is completed using MySQL whilst PHP is used to connect to the database back end and provide a webpage containing JSON code that can be interpreted and parsed by the Android application to add extra information to the existing activities.

During the implementation, various issues were encountered. These issues were mostly caused by inexperience with working in the Android development environment. The solutions found as well as the problems experienced have been documented.

7

Testing and Evaluation

7.1 Introduction

This section will describe, in full, the testing performed on the application both during and post implementation as well as a description of the testing methods adopted and the resulting test logs in full. Any bugs or unexpected results will be documented and an analysis of the cause will be provided as well as any potential solutions to the issue. The final applications will then be evaluated and presented alongside results of a user satisfaction survey.

7.2 Testing Methodology

The entire project was tested thoroughly throughout the implementation stage. Using an agile methodology meant that once testing had been completed, should the tested section of the project not meet the requirements, it was possible to return to previous stages of the software development life cycle to continue development before release. Returning back in this way is known as an 'iteration' and meant the design or implementation of the software was changed before re-testing and allowed for 'release' as the finished product.

Through this methodology, the Map View and Speed Test activities required 2 iterations whilst the Mobile database required a single iteration.

Testing was initially performed using the Android Emulator and allowed for functionality to be tested as well as to observe what was happening behind the scenes using Android's debugging tools in a form of 'white-box' testing. Debugging the application in this way was achieved through the use of 'logcat', Android's logging system that provided a mechanism for collecting and viewing system debug output from the emulator running the early versions of the app [59]. Logcat provided detailed output to any errors and locations of the code that caused them, allowing changes to be made to any code quickly and effectively.

Using the Android Emulator also allowed for multiple prototypes to be mocked up without taking up room on a physical device. These prototypes allowed for separate components of a larger structure to be tested individually, such as the graph view. This allowed effective testing as it ensured the components were individually functional before bringing them together for the implementation of the overall UI design.

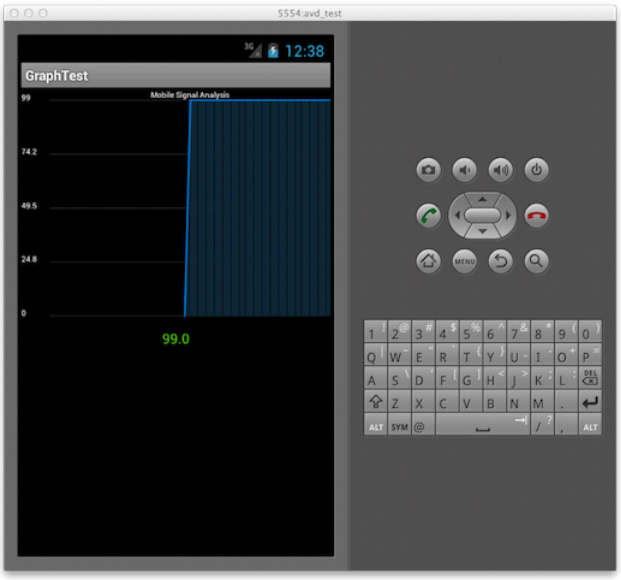


Figure 7.1: **An early Graph prototype running on the Android Emulator**

Further testing was completed later in the project using a physical device. This allowed for 'black-box' testing to be performed through looking at the functionality of the application without looking at code and 'inner workings' of the application. This approach is also used for acceptance testing, which looks at the software requirements and tests whether the requirements have been met. The device used was a Samsung SIII, a popular high end smart phone, with the following specifications:

Table 7.1: **Samsung SIII Spec**

Attribute	**Samsung SIII [60]**
Operating System	Android 4.1.2 (Jelly Bean)
CPU	Quad-Core 1.4GHz
RAM	1 GB
Internal Memory	32 GB
Display	720 x 1280

This allowed the app to be tested on the type of phone it was marketed for and allowed the application to be used with real data rather than working with the emulator.

During testing, logs were made of the overall outcome of each test. The test logs for the entire system are documented in Appendices A and B.

7.2.1 Problems Encountered

Through the thorough testing of the application it was possible to find a number of previously unrecognised and unencountered bugs. These bugs were small and, due to time constraints, were not patched. However, they were not picked up during user feedback and do not pose a great risk of causing the application to crash.

There were multiple bugs found with the use of the Map Overlays within the Map View interface. These overlays appear in the form of 'clickable' images placed on top of the Google Map interface and are placed on the map during changes of location and on requesting mobile mast overlays.

Whilst testing it was observed that during the period in which GPS navigation is being enabled, the accuracy of the GPS location can vary significantly and place overlays in incorrect geographical locations. This is due to the GPS attempting to locate the user and being able to track to a higher degree of accuracy once the connection has been made over a longer period of time.

Further to this bug, another is observable should there be multiple overlays on screen and the user chooses to zoom out. During this change of perspective the overlays, currently displayed on the map view, become increasingly bunched together causing it to be harder to select individual overlays to access the information associated with them. This is due to the overlays and their images remaining at their given location despite the map area for that location becoming smaller. This could be fixed through removing overlays once the user reaches a certain zoom level.

Finally, should a user request to view mobile base station information in the form of overlays on the Map View, before requesting the same information again, the overlays will form on top of each other. This can cause large amounts of overlays to be displayed incorrectly when they should not have been placed. This is due to the application currently not checking the previous location of the request for the mobile base-station data and could be patched through checking whether the distance between where a current request has been made and the location of the last request is beyond a threshold.

7.2.2 Summary

Overall from the testing of the system through the use of 'white-box' and 'black-box' testing, it has been possible to see that, on the whole, the application works well and meets the system requirements specification. Furthermore, only small and negligible bugs were found with the system during the testing of the application. This could be partially due to the choice of testing methodology as well as the agile methodology adopted for the project. The agile methodology

provided time to spot bugs during testing and complete iterations to factor them out through changing the code accordingly.

These previous bugs and issues are described in Section 6 in detail.

7.3 User Feedback

To fully evaluate the project, the application should be thoroughly reviewed by potential users of the application. To receive this user feedback, a short questionnaire was produced to provide an insight into what users of the application felt about the app, how important they felt the principles behind it were and whether they would consider using the app on their smart phone.

7.3.1 Questionnaire Design

The questionnaire was spilt into two sections, the first of which contained closed questions about the background of the user and were asked to be completed **before** the users began to use the application. The questions asked within the first section are as follows:

- **Student ID (optional), Age, Gender**
  It should be noted that on distributing the questionnaires, users were told that providing their Student ID was optional - These IDs have been removed from the questionnaires to protect the anonymity of the participants
- **Do you currently own a smart phone?**
- **If yes, what operating system does your phone use?**
- **How important would you class 'good' Mobile Phone signal strength in terms of your day-to-day phone usage?**
- **How important would you class 'good' Mobile Phone signal speed in terms of your day-to-day phone usage?**
- **Would you be interested in using a mobile application that can provide real-time details about your phone's received signal strength and speed?**

These questions were asked to gauge whether or not the creation and potential release of the application would be welcomed. These statistics would be deemed useful as it would allow a rough estimate to how popular and desired an application similar to the one produced is in the current market.

Further to this, it also provides an approximation of the perceived importance that users regard their mobile signal strength and speed in the context of their own phone usage. These statistics are important as they look more in depth at the target audience and analyse whether there is a common theme and if many users find these criteria important in day to day life.

The second part of the questionnaire was given to the participants after they had used the application. This part of the questionnaire focused primarily on the user experience and the following open questions were asked:

- **Did you find the application easy to use?**
- **What feature(s) of the application did you like?**
- **What feature(s) of the application did you think could be improved?**
- **Would you consider this application 'useful' in your day-to-day life? Please explain your answer.**
- **Do you have any other comments or suggestions about the application?**

These questions were asked to gauge the overall user experience; To see what the target audience of the app think the project does well and to see what they believed requires improvement. This information is helpful as it gives an overview to how potential users currently view the app and whether the project has created something that would be accepted by users.

Ten of these questionnaires were produced and handed out to willing students of the University of Nottingham. From these distributed questionnaires, there were five responses. Copies of which can be viewed in Appendix C.

7.3.2 Results

From the results from the questionnaire the following analysis can be made.

All participants currently owned a smart phone either running on Apple's iOS or Android. Those with iOS devices expressed verbally that they were familiar with the Android platform and it's correct use in addition to that of their own devices.

Participants were asked about how 'important' they felt that mobile signal strength and the speed of their connection is, in terms of their day to day phone usage.

The outcome of this question was as follows:

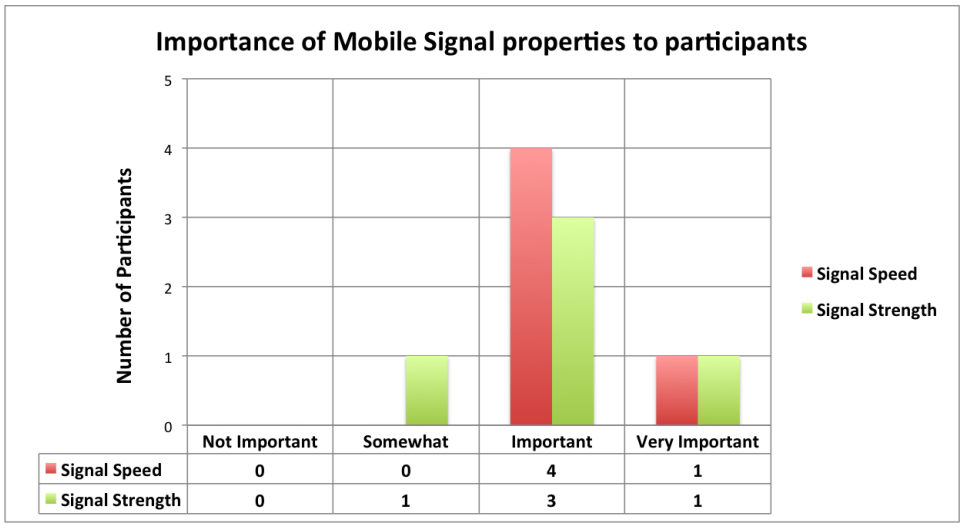


Figure 7.2: **Importance of Mobile Signal to the participants of the user Questionnaire**

As Figure 7.2 shows, 90% of participants found both signal strength *and* speed either important or very important in their day to day phone usage. This information, despite being gathered from a small sample, suggests that many potential users of the application are interested in the fundamental principles behind the application. Some participants also provided verbal feedback in their choices, stating that they use their smart phones 'as mobile computers' and required good signal and a fast connection to 'stay connected to Facebook, Twitter and e-mail'.

From these answers, the participants were then asked whether they were interested in an application that could provide real time information about these criteria. This produced the following results:

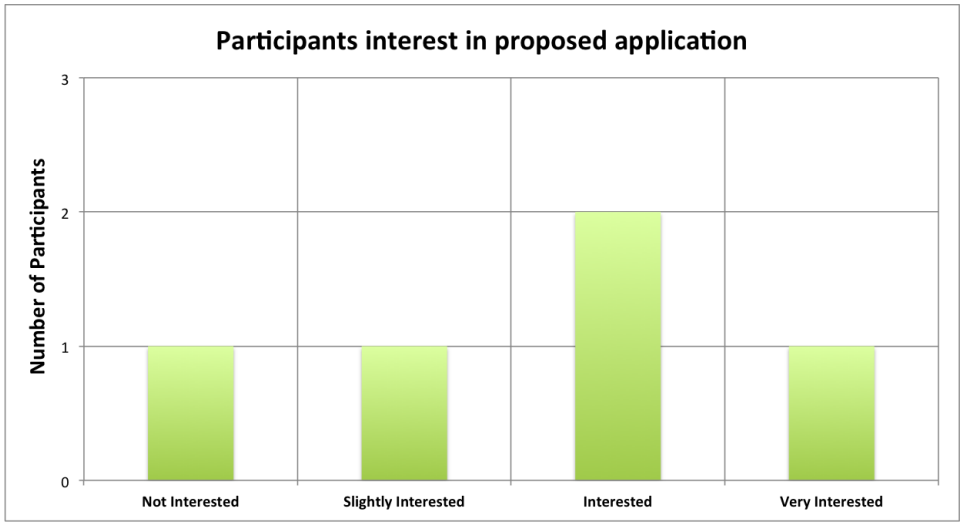


Figure 7.3: **Participant Interest In Proposed Application**

Despite nearly all participants feeling that the two mobile signal properties were ‘important’, there was a much larger spread in opinion regarding the participants interest in using a mobile application to find information about their device’s mobile signal. Just over half of participants stated they would be interested in using such an application.

The participants were then allowed to interact with the produced application. No assistance was given unless it was requested and users were asked to fill in the final section of the questionnaire once they felt they had used the application significantly and had formed an opinion on it’s features. The questions within this section were all ‘subjective’ questions that participants could provide their opinions on the overall experience. It was anticipated that, through these open ended questions, users would provide some useful feedback and well formed opinions on the application, it’s user interfaces and functions.

All five participants found the application easy to use and intuitive. One participant commented further stating ‘everything [is] clear and well labelled’. This was one of the requirements set by the software requirements specification and has been successfully achieved judging on the feedback from users.

Participants were then asked to name the features of the application they liked. These results varied but could be categorised as two participants specified the ‘Map View’ as their favourite feature of the application. Both these users expressed verbally that they considered this feature to be ‘cool’ and ‘different’ and that they had not seen an application with an interface similar to that of the Map View before. Another participant stated that the Speed Test was their personal favourite element of the application. They stated verbally they felt its ‘accuracy was awesome’ and that ‘could be quite useful’. One participant also mentioned the ‘live graphs’ as a particular favourite.

These results show overall users found multiple features engaging and interesting. This feedback is positive as it suggests that the application as a whole is interesting to users and that they enjoyed using these particular features of the application.

However, when asked about what features the participants would change about the application, 80% mentioned the Speed Test interface. Verbal feedback for this question varied from ‘the interface is a bit plain’ to ‘needs more information about speeds’. Participants seemed to like the the basics behind the interface but felt the UI was not as good as that of the ‘Map View’.

From these results it is clear that the UI of the Speed Test could be improved and that users didn’t ‘dislike’ it as such, but felt it could be improved. Should the project continue, this is an element that may require further work.

The following question asked participants, after their experience with the

application, would they consider using this app on a day to day basis to help with their phone usage. Again, this provoked a mixed response from participants. Three participants specifically stated that they **would** use the application in their day to day life. These participants stated that they were 'impressed with the application' and felt that it could 'often be useful'. One participant stated that they would 'possibly' use the application but it would depend on whether the application 'was free [to download]'. Finally, the last participant felt that they would use the application but 'not day to day, but if [they] wanted to download something or watch something on the go'.

These results can be seen as extremely positive. All participants stated that, in multiple ways, they would all be prepared to use the application further should it be released for their device. This is a huge success and gives a very positive outlook for the application should it continue development.

Finally, users were asked if they wished to make any further comments on the application. Only one participant answered this, stating they felt the application was 'really cool'.

7.3.3 Summary of User Feedback

On the whole, feedback from users was relatively positive. Users seemed to enjoy using the application and found the information it provided appropriate and informative. Many of the users provided some good constructive feedback regarding the app and this has been taken into consideration for further work, should the project continue.

One criticism of the questionnaire should be the number of responses received. Due to time constraints, it was not possible to collect more than five out of ten distributed questionnaires. This was slightly disappointing and made analysis of the results harder to backup as being a general trend instead of anomalies.

The questionnaire was appropriate but perhaps could have asked further questions such as individual questions about each of the functions of the application. This would have provided more clear cut constructive feedback and may have been slightly more beneficial in terms of judging the overall feedback from participants.

8

Further Work and Summary

This section details further developments that could be made to the application and overall project should it be continued. A critical appraisal of the project and a personal reflection are also included to complete this review.

8.1 Future Directions

The following subheadings describe possible routes and improvements to the application that could be made should this project continue. Many of these were not completed due to constraints with the project such as time, cost or lack of technical knowledge.

8.1.1 Speed Test Interface

Further work on the Speed Test could include a full redesign of the UI as many participants of the questionnaire believed it could be improved. Further changes could include the ability to also test upload speeds of the current connection, the ability to keep track of previous speed test results and the ability to compare these results with friends on popular social media sites such as Facebook or Twitter. These would benefit the application by allowing the user to access further information about their device's signal whilst also adding the social element to the application.

8.1.2 Repeating Questionnaire

The study undertaken could be repeated with more participants in hope that these would verify the results shown in the previous study. Furthermore, a redesign of the survey may also help provide better and more constructive feedback.

8.1.3 Long Term Use

A study that focuses on the long term use of the system could potentially be designed to examine how, why and when users use the application. A study such as this would provide an insight into user habits and allow for the application to take these into consideration during its continued development.

8.1.4 Different Mobile Platforms

Currently the application is only designed for newer Android devices. In the future it may be worth considering releasing the application on other mobile operating systems such as Apple's iOS or Blackberry OS. This would be beneficial as this drastically increases the number of potential users.

8.1.5 Release Application

Releasing the application on an app store such as Google Play, may be worth considering in the future. This would allow the general public to use the application created during the project. This may be beneficial as it could bring attention to the app and possibly become quite popular. It would also allow for more testing to occur as users could report feedback on any issues they encounter or allow them to provide positive comments on aspects of the application they like.

8.2 Concluding Remarks

The aims and objectives of the project have been completed in full. A mobile application has successfully been developed that accurately and intuitively provides users with real time data about their current mobile device's connection to the mobile network. This application has been thoroughly tested and evaluated by users.

The motivation behind the project was discussed in full along with the aims and objectives the project set out to achieve followed by an analysis of possible causes of mobile 'black spots' that were discussed in depth in conjunction with history of mobile signal technologies and protocols, to determine the background behind the creation of this application.

A study into relevant systems was conducted and each relevant system was analysed and evaluated before a proposal for the system was formulated.

A number of designs for the app and it's multiple interfaces were then explored and evaluated before a full requirements specification was produced. The

application was then implemented and tested thoroughly.

Some aspects of the project were not addressed as fully as was initially hoped. Measuring the potential global use of the application was not possible due to time constraints as the application was only tested within the UK.

8.3 Critical Appraisal

Overall the project went very well and fulfilled the aims and objectives set out during the creation of the project. An Android application was successfully created to monitor live mobile signal properties, the methods used to create it were explored and the overall project evaluated.

The project took a different path than originally intended and provided extra functionality through the creation of the mobile base station database. However due to the dataset recently becoming 'out of date' and the uncertainty regarding its future [34], it is not clear how beneficial this additional feature will be in the long term.

The Map View interface meets the software specifications, as described earlier in the documentation, to a high standard. The interface looks modern and professional, and received positive feedback from users. The design of the interface gives the application a 'unique selling point' and exceeded expectations.

The Speed Test interface, despite meeting the design specification, does not look as professional as it possibly could have been. The interface feels slightly underwhelming considering the high standard of other views such as the Map View and had some criticisms from the user feedback regarding this. This interface was recommended for further work for these reasons.

8.4 Personal Reflection

On the whole I believe the project was a success. I learnt to program for the Android platform from scratch and produced an application I would consider 'useful' that incorporated interesting and novel features I had not seen in an application before. I feel that during the project, my increased skills and confidence in using Android led the project to go further than initially anticipated and this is something I am very proud of.

I have also learnt a lot about time management and working using an agile methodology. I believe that my time management could have been better and should I repeat the project again, I would attempt to complete sections of this document whilst coinciding with physical work for the project.

I thoroughly enjoyed producing the application and would certainly consider extending this dissertation and the further work as previously discussed.

Bibliography

[1] MobiThinking.com (2012) *Global mobile statistics 2012 Part A: Mobile subscribers...* [online] Available at: `http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a#subscribers` [Accessed: 25 Nov 2012]

[2] Apple.com (2012) *Apple - Press Info - Apple's App Store Downloads Top 25 Billion.* [online] Available at: `http://www.apple.com/pr/library/2012/03/05Apples-App-Store-Downloads-Top-25-Billion.html` [Accessed: 29 Dec 2012].

[3] Official Android Blog (2012) *Google Play hits 25 billion downloads.* [online] Available at: `http://officialandroid.blogspot.co.uk/2012/09/google-play-hits-25-billion-downloads.html` [Accessed: 29 Dec 2012].

[4] Android (2012) *Google Play App Installs.* [image online] Available at: `http://bit.ly/GooglePlay25img` [Accessed: 29 Dec 2012].

[5] Ofcom (2012) *Ofcom unveils plans to avoid mobile 'capacity crunch'.* [online] Available at: `http://media.ofcom.org.uk/2012/11/16/ofcom-unveils-plans-to-avoid-mobile-capacity-crunch/` [Accessed: 30 Dec 2012].

[6] Worldtimezone.com (2013) *GSM World Coverage Map- GSM Country List by frequency bands.* [online] Available at: `http://www.worldtimezone.com/gsm.html` [Accessed: 24 Jan 2013].

[7] developer.android.com (n.d.) *Android, the world's most popular mobile platform — Android Developers.* [online] Available at: `http://developer.android.com/about/index.html` [Accessed: 27 Jan 2013].

[8] Play.google.com (2013) *Google Accounts.* [online] Available at: `https://play.google.com/apps/publish/v2/signup/` [Accessed: 1 May 2013].

[9] Felker, D. (2011) *Android Application Development.* Indianapolis: Wiley Publishing, p.13.

[10] GSMWorld (2012) *History.* [online] Available at: `http://www.gsmworld.com/aboutus/history` [Accessed: 27 Jan 2013].

[11] Poole, I. (2006) *Cellular Communications Explained: From Basics to 3G*. Elsevier, p.6-10.

[12] EE (2012) *Network Evolution*. [online] Available at: `http://explore.ee.co.uk/network-evolution` [Accessed: 6 May 2013].

[13] UK Mobile Coverage (2011) *Which Mobile Phone Network Has The Best Signal Coverage?*. [online] Available at: `http://ukmobilecoverage.co.uk/best` [Accessed: 29 Dec 2012].

[14] Mongabay.com (2008) *List of Countries by Land Mass* [Ranked by Area]. [online] Available at: `http://www.mongabay.com/igapo/world_statistics_by_area.htm` [Accessed: 8 Mar 2013].

[15] Orange.co.ke (n.d.) *Orange Coverage*. [online] Available at: `http://orange.co.ke/map/index.html` [Accessed: 9 Mar 2013].

[16] Geonames.org (n.d.) *Biggest Cities Kenya*. [online] Available at: `http://www.geonames.org/KE/largest-cities-in-kenya.html` [Accessed: 9 Mar 2013].

[17] Indexmundi.com (2012) *Kenya Population - Demographics*. [online] Available at: `http://www.indexmundi.com/kenya/population.html` [Accessed: 9 Mar 2013].

[18] BBC News (2012) *O2 suffers mobile call problems*. [online] Available at: `http://www.bbc.co.uk/news/technology-19928507` [Accessed: 9 Mar 2013].

[19] Halliday, J. (2012) *O2 network 'fully restored' after 24-hour blackout*. [online] Available at: `http://www.guardian.co.uk/technology/2012/jul/12/o2-outage-turn-off-3g-data` [Accessed: 9 Mar 2013].

[20] Wikipedia.org (2012) *Fresnel Zone Image*. [image online] Available at: `http://en.wikipedia.org/wiki/File:FresnelSVG1.svg`. [Accessed: 9 Mar 2013].

[21] ZyTrax.com (2008) *ZyTrax - Fresnel Zones and their Effect*. [online] Available at: `http://web.archive.org/web/20060328012647/http://www.zytrax.com/tech/wireless/fresnel.htm` [Accessed: 9 Mar 2013].

[22] Poole, I. (2006) *Cellular Communications Explained: From Basics to 3G*. Elsevier, p.20.

[23] Cellphones.org (n.d.) *Why Your Cell Reception Sucks*. [image online] Available at: `http://cellphones.org/blog/cellular-phone-reception/`.

[24] Opensignal.com (2012) *OpenSignal Android App*. [online] Available at: `http://opensignal.com/android/` [Accessed: 22 Apr 2013].

[25] Play.google.com (2013)OpenSignal-AndroidAppsonGooglePlay. [online] Available at: https://play.google.com/store/apps/details?id=com.staircase3.opensignal&hl=en

[26] Openbmap.org (n.d.) *Wireless Collaborative Map*. [online] Available at: http://openbmap.org/ [Accessed: 22 Apr 2013].

[27] Hutchison, L. (2010) *metalev: Android vs iPhone 4 signal strength display (FWIW)*. [online] Available at: http://www.metalev.org/2010/07/android-vs-iphone-4-signal-strength.html [Accessed: 22 Apr 2013].

[28] Anandtech (2010) *Signal Bar Mappings*. [image online] Available at: http://images.anandtech.com/reviews/gadgets/apple/iPhone4/part2/signalbarmapping.jpg [Accessed: 23 Apr 2013].

[29] Chinn, H. A., Gannett, D. K., & Morris, R. M. (1940). *A new standard volume indicator and reference level*. Proceedings of the IRE, 28(1), p1-17.

[30] Lifehacker (2013) *See the Actual Signal Strength on Your iPhone or Android*. [online] Available at: http://lifehacker.com/5929546/see-the-actual-signal-strength-on-your-iphone-with-this-quick-tweak [Accessed: 22 Apr 2013].

[31] Apple.com (2010) *Apple Press Info - Letter from Apple Regarding iPhone 4*. [online] Available at:http://www.apple.com/uk/pr/library/2010/07/02Letter-from-Apple-Regarding-iPhone-4.html [Accessed: 22 Apr 2013].

[32] Opensignal.com (2012) *OpenSignal*. [online] Available at: http://opensignal.com/ [Accessed: 22 Apr 2013].

[33] Sitefinder.ofcom.org.uk (2013) *'Sitefinder' Mobile Phone Base Station Database.* [online] Available at: http://www.sitefinder.ofcom.org.uk/ [Accessed: 19 Apr 2013].

[34] Stakeholders.ofcom.org.uk (2008) *Ofcom — Sitefinder dataset*. [online] Available at: http://stakeholders.ofcom.org.uk/sitefinder/sitefinder-dataset/ [Accessed: 19 Apr 2013].

[35] Telecomsnetworks.com (2013) *Telecoms Networks*. [online] Available at: http://www.telecomsnetworks.com/ [Accessed: 20 Apr 2013].

[36] IEEE, *Guide to Software Requirements Specifications* (1984), IEEE Std 830-1984 Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=278253&isnumber=6883

[37] Developer.android.com (2012) *Activity — Android Developers*. [online] Available at: http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle [Accessed: 28 Apr 2013].

[38] Gartner.com (2012) G*artner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent*. [online] Available at: `http://www.gartner.com/newsroom/id/2237315` [Accessed: 27 Apr 2013].

[39] Developers.google.com (2013) *Google Maps Android API v2 - Google Developers*. [online] Available at: `https://developers.google.com/maps/documentation/android/` [Accessed: 27 Apr 2013].

[40] Developers.google.com (n.d.) *Google Maps Android API v2 Documentation - Overlay*. [online] Available at: `https://developers.google.com/maps/documentation/android/v1/reference/` [Accessed: 27 Apr 2013].

[41] Jjoe64.com (2012) *GraphView Library*. [online] Available at: `http://www.jjoe64.com/p/graphview-library.html` [Accessed: 27 Apr 2013].

[42] Apache.org (2012) *org.apache.http (HttpCore 4.2.4 API)*. [online] Available at: `http://hc.apache.org/httpcomponents-core-ga/httpcore/apidocs/org/apache/http/package-summary.html` [Accessed: 27 Apr 2013].

[43] Json.org (n.d.) *JSON in Java*. [online] Available at: `http://json.org/java/` [Accessed: 27 Apr 2013].

[44] phpmyadmin.net (2013) *phpMyAdmin*. [online] Available at: `http://www.phpmyadmin.net/home_page/index.php` [Accessed: 27 Apr 2013].

[45] Developer.android.com (2012) *Managing Virtual Devices — Android Developers*. [online] Available at: `http://developer.android.com/tools/devices/index.html` [Accessed: 27 Apr 2013].

[46] Developer.android.com (2012) *Android Emulator — Android Developers*. [online] Available at: `http://developer.android.com/tools/help/emulator.html` [Accessed: 27 Apr 2013].

[47] Developer.android.com (2012) *LocationListener — Android Developers*. [online] Available at: `http://developer.android.com/reference/android/location/LocationListener.html` [Accessed: 28 Apr 2013].

[48] Developer.android.com (n.d.) *PhoneStateListener — Android Developers*. [online] Available at: `http://developer.android.com/reference/android/telephony/PhoneStateListener.html#onSignalStrengthsChanged(android.telephony.SignalStrength)` [Accessed: 28 Apr 2013].

[49] ETSI, TS. 127 007 V8. 5.0 (2008-10) (2010) *Digital cellular telecommunications system (Phase2+) Universal Mobile Telecommunications System (UMTS)*. p 81. Available at: `http://www.etsi.org/deliver/etsi_ts/127000_127099/127007/08.05.00_60/ts_127007v080500p.pdf`

[50] Developer.android.com (2012) *SignalStrength — Android Developers*. [online] Available at: `http://developer.android.com/reference/android/telephony/SignalStrength.html#getGsmSignalStrength()` [Accessed: 28 Apr 2013].

[51] Android-er.blogspot.co.uk (2011) *Android-er: Use Handler and smart phone to generate a periodic event*. [online] Available at: `http://android-er.blogspot.co.uk/2011/09/use-handler-and-runnable-to-generate.html` [Accessed: 28 Apr 2013].

[52] AndroidPit (2013) *Shake*. [image online] Available at: `http://fs02.androidpit.info/userfiles/44704/image/Bilder/Samsung/Galaxy%20Note/Bewegungen/galaxy%20note%20schuetteln.jpg` [Accessed: 28 Apr 2013].

[53] Developer.android.com (2012) *Sensors Overview — Android Developers*. [online] Available at: `http://developer.android.com/guide/topics/sensors/sensors_overview.html` [Accessed: 28 Apr 2013].

[54] Developer.android.com (2012) *SensorEventListener — Android Developers*. [online] Available at: `http://developer.android.com/reference/android/hardware/SensorEventListener.html` [Accessed: 28 Apr 2013].

[55] Developer.apple.com (2013) *dd (man page)*. [online] Available at: `https://developer.apple.com/library/mac/#documentation/Darwin/Reference/Manpages/man1/dd.1.html` [Accessed: 30 Apr 2013].

[56] Crossley, M. (1999). *A guide to coordinate systems in Great Britain. Ordnance Survey. Southampton* Available at: `http://www.gps.gov.uk/additionalInfo/images/A_guide_to_coord.pdf` [Accessed: 28 Apr 2013].

[57] Jstott.me.uk (2006) *OSRef*. [online] Available at: `http://www.jstott.me.uk/jcoord/api/1.0/uk/me/jstott/jcoord/OSRef.html#toLatLng()` [Accessed: 29 Apr 2013].

[58] Php.net (2013) *PHP: json_encode - Manual*. [online] Available at: `http://php.net/manual/en/function.json-encode.php` [Accessed: 29 Apr 2013].

[59] Developer.android.com (2012) *logcat — Android Developers*. [online] Available at: `http://developer.android.com/tools/help/logcat.html` [Accessed: 1 May 2013].

[60] Gsmarena.com (2012) *Samsung I9300 Galaxy S III - Full phone specifications*. [online] Available at: `http://www.gsmarena.com/samsung_i9300_galaxy_s_iii-4238.php` [Accessed: 1 May 2013].

Appendix A

Test Logs - Android

The following table shows the results of final testing of the Android app and were completed in the Android Simulator and on a Samsung Galaxy SIII.

Function	**Input**	**Result**	**Modifications & Comments**
MAP VIEW			
Finding location from GPS	None (Automatic)	Success, location found and to high level of accuracy	None
The system must display the signal strength in dBm	None (Automatic)	Success, shown on Location dialog	Live graph data is shown in ETSI standard
Request user turns on GPS should it not be on	None (Automatic)	Success, the user is taken to the device's Location settings and a message prompting them to turn on GPS is given	After testing, this feature was also used for the Mast Database's Text View activity as it also requires GPS
Map should locate user and place an Overlay at location	None (Automatic)	Success, an overlay is placed at the correct current location	On very rare occasions, multiple overlays may be placed slightly out from current location whilst the GPS service locates the user. This may be required for future work.
Overlays should provide information once tapped	User taps on icon	Success, accurate data displayed for correct overlays	None

The map should move to the latest known location	None (Automatic)	Success, map displays correctly and moves to current location	Note: The more the user is zoomed out, the less this change is noticeable
Zoom controls should react correctly to user input, zooming 'in' or 'out'	User taps on zoom icons	Success, zoom controls work as expected	On zooming out, any Overlays become increasingly bunched and harder to tap on each individual Overlay. This bug may be required for further work.
Pinch gestures should alter the Map's level of zoom	User pinches screen	Success, the map reacts to gestures made by the user and zooms accordingly	None
Menu items should display on top bar should the device be wide enough	None (Automatic)	Success, the three menu items are displayed with their icons	If there is not enough space, the items can be accessed through the 'menu key' on the device
Tapping on the Satellite menu item should put the map in Satellite View	User taps on menu icon	Success, the map is put into the correct view	This has no effect should the map already be in this view
Tapping on the Road menu item should put the map in Road View	User taps on menu icon	Success, the map is put into the correct view	This has no effect should the map already be in this view
Tapping on the Antenna menu item should load the location of Mobile Basestations to the map as overlays	User taps on icon	Success, the overlays are added to the screen	Tapping on the icon after already tapping previously causes the overlays to be added again. This bug could be something to address in further work
A 'Loading' screen is displayed whilst data is accessed from the database	None (Automatic)	Success, 'spinner' is shown during download and dismissed once complete	None

The icon on the Location Overlay dialog should reflect signal strength	None (Automatic)	Success, icon changes depending on relative strength as defined by ETSI [49]	None
Tapping on the Mast Overlay should return accurate information about that mast	User taps on Overlay	Success, dialog box is shown on screen with relevant accurate information about the mast and the approximate distance of that mast from the user	If there are multiple masts at a location, only one of the dialog boxes is avaiable. This may require further work.
The graph view should provide real-time signal strength information	None (Automatic)	Failure, the graph does not provide live results until it is interacted with. The graph will then display the correct data live	None, this bug is known and has been discussed previously, including methods to fix this issue. This will feature in further work.
SPEED TEST VIEW			
The speed test button should begin a speed test	On click of the Test button	Success, test begins instantly	None
During a Speed Test, the progress bar should update accordingly	None (Automatic)	Success, test progress is monitored accurately and shown correctly by progress bar	Previous issue had seen the bar stuck at 99%. This was fixed before acceptance testing
During a Speed Test, the text view should update accordingly	None (Automatic)	Success, test progress is monitored accurately and shown correctly by the text view	None
The graph view should provide real-time download speed information	None (Automatic)	Failure, the graph does not provide live results until it is interacted with. The graph will then display the correct data live	None, this bug is known and has been discussed previously, including methods to fix this issue. This will feature in further work.
Once complete, the Text View should display results within the Text View	None (Automatic)	Success. The Text View is updated with relevant information once test is completed	None

When the focus is moved for the Activity, any current Speed test should be cancelled	None (Automatic)	Failure. Tests remained running in background	Code was modified so on exiting the application or moving focus from the Speed Test activity, the download would stop. This bug is now fixed
MAST DATABASE TEXT VIEW			
The text view should be updated with mast information once the Button has been clicked	User taps Button	Success, mast information is displayed in the text view	None
A 'Loading' screen is displayed whilst data is accessed from the database	None (Automatic)	Success, 'spinner' is shown during download and dismissed once complete	None
ABOUT DIALOG			
Information about the app should be displayed when the user navigates to the About Dialog	None (Automatic)	Success, the about text is shown	None
MAIN MENU			
Clicking on each button takes users to the correct activity	User taps button	Success for all buttons	None

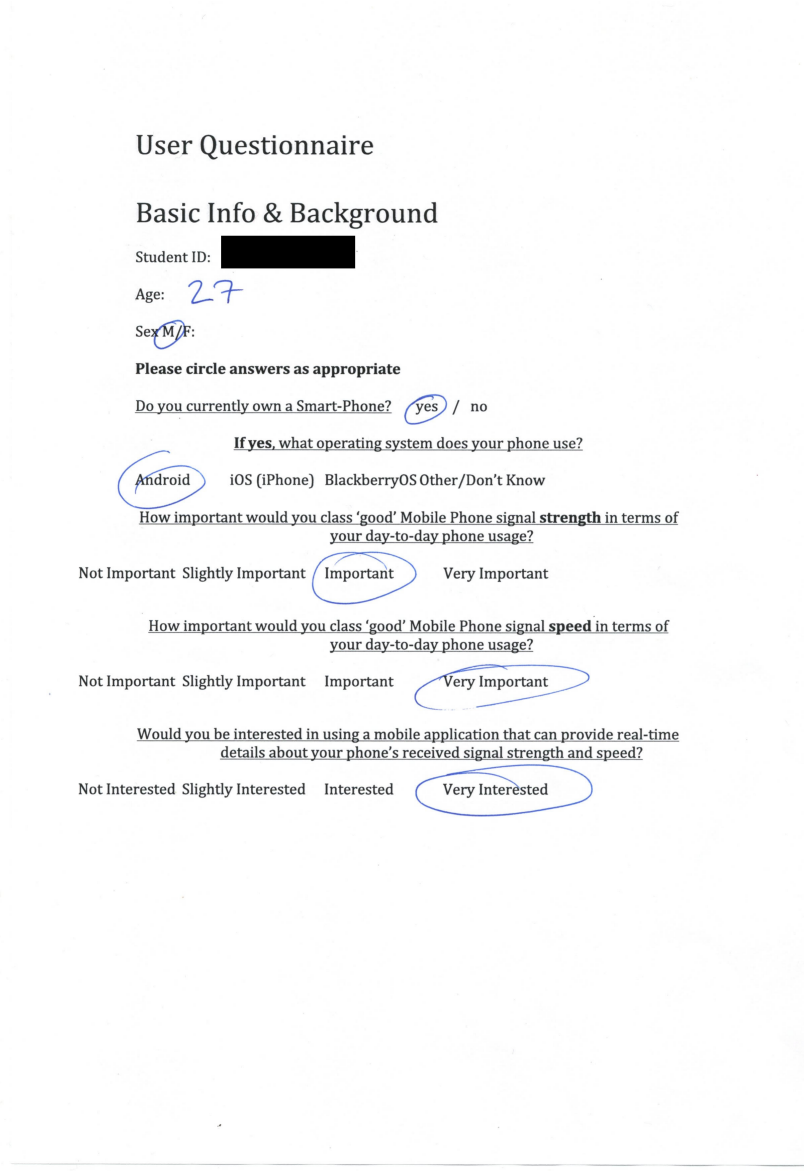
Appendix B

Test Logs - Database Backend

MAST DATABASE	**Input**	**Result**	**Modifications & Comments**
Providing a valid input URL produces the correct output	None (Automatic)	Success, the PHP produced valid and correct JSON.	URL: json.php?lat=52.94568293 &lon=-1.17908524 &dif=0.0125
Providing an invalid input URL produces an empty list and no errors	None (Automatic)	Success, the PHP produced valid and correct empty JSON {"Locations": []}.	URL: jjson.php?lat=test &lon=test
Changing the `diff` variable within the URL should regulate the difference in latitude and longitude the query executes	None (Automatic)	Success, more/less masts are displayed depending on the size of the area covered and changing value of `diff`.	None
Changing the `debug` variable to `true` within the URL should provide information about the database results returned	None (Automatic)	Success, the number of rows returned and the total time taken to execute the MySQL query are both displayed.	None

Appendix C

User Questionnaire Feedback



User Questionnaire

Basic Info & Background

Student ID: [redacted]

Age: 27

Sex M/F:

Please circle answers as appropriate

Do you currently own a Smart-Phone? yes / no

If yes, what operating system does your phone use?

Android iOS (iPhone) BlackberryOS Other/Don't Know

How important would you class 'good' Mobile Phone signal **strength** in terms of your day-to-day phone usage?

Not Important Slightly Important Important Very Important

How important would you class 'good' Mobile Phone signal **speed** in terms of your day-to-day phone usage?

Not Important Slightly Important Important Very Important

Would you be interested in using a mobile application that can provide real-time details about your phone's received signal strength and speed?

Not Interested Slightly Interested Interested Very Interested

Interface Questionnaire

Please write answers as appropriate

Did you find the application easy to use?

Yes

What feature(s) of the application did you like?

All of them.

What feature(s) of the application did you think could be improved?

None.

Would you consider this application 'useful' in your day-to-day life? Please explain your answer:

Yes.

Do you have any other comments or suggestions about the application?

No

Thank you for your participation

User Questionnaire

Basic Info & Background

Student ID: [redacted]

Age: 24

Sex M/F: M

Please circle answers as appropriate

Do you currently own a Smart-Phone? (yes) / no

If yes, what operating system does your phone use?

(Android) iOS (iPhone) BlackberryOS Other/Don't Know

How important would you class 'good' Mobile Phone signal **strength** in terms of your day-to-day phone usage?

Not Important (Slightly Important) Important Very Important

How important would you class 'good' Mobile Phone signal **speed** in terms of your day-to-day phone usage?

Not Important Slightly Important (Important) Very Important

Would you be interested in using a mobile application that can provide real-time details about your phone's received signal strength and speed?

Not Interested Slightly Interested (Interested) Very Interested

Interface Questionnaire

Please write answers as appropriate

Did you find the application easy to use?

Yes everything ~~clear~~ clear and well labdal

What feature(s) of the application did you like?

Signal Map

What feature(s) of the application did you think could be improved?

Speed test interface could be improved.

Would you consider this application 'useful' in your day-to-day life? Please explain your answer:

Yes interesting to compare locations data

Do you have any other comments or suggestions about the application?

Thank you for your participation

User Questionnaire

Basic Info & Background

Student ID: [redacted]

Age: 20

Sex M/F: M

Please circle answers as appropriate

Do you currently own a Smart-Phone? (yes) / no

If yes, what operating system does your phone use?

(Android) iOS (iPhone) BlackberryOS Other/Don't Know

How important would you class 'good' Mobile Phone signal strength in terms of your day-to-day phone usage?

Not Important Slightly Important (Important) Very Important

How important would you class 'good' Mobile Phone signal speed in terms of your day-to-day phone usage?

Not Important Slightly Important (Important) Very Important

Would you be interested in using a mobile application that can provide real-time details about your phone's received signal strength and speed?

Not Interested Slightly Interested (Interested) Very Interested

Interface Questionnaire

Please write answers as appropriate

Did you find the application easy to use?

Yes, really simple!

What feature(s) of the application did you like?

live graphs

What feature(s) of the application did you think could be improved?

Speed test interface needs more

Would you consider this application 'useful' in your day-to-day life? Please explain your answer:

Possibly, would depend on if app was free?

Do you have any other comments or suggestions about the application?

no.

Thank you for your participation

User Questionnaire

Basic Info & Background

Student ID: [redacted]

Age: 21

Sex M/F: F

Please circle answers as appropriate

Do you currently own a Smart-Phone? (yes) / no

If yes, what operating system does your phone use?

Android (iOS (iPhone)) BlackberryOS Other/Don't Know

How important would you class 'good' Mobile Phone signal **strength** in terms of your day-to-day phone usage?

Not Important Slightly Important (Important) Very Important

How important would you class 'good' Mobile Phone signal **speed** in terms of your day-to-day phone usage?

Not Important Slightly Important (Important) Very Important

Would you be interested in using a mobile application that can provide real-time details about your phone's received signal strength and speed?

(Not Interested) Slightly Interested Interested Very Interested

Interface Questionnaire

Please write answers as appropriate

Did you find the application easy to use?

Yes

What feature(s) of the application did you like?

The map view

What feature(s) of the application did you think could be improved?

speed test interface

Would you consider this application 'useful' in your day-to-day life? Please explain your answer:

Yes

Do you have any other comments or suggestions about the application?

It's really cool

Thank you for your participation

User Questionnaire

Basic Info & Background

Student ID: [redacted]

Age: 21

Sex M/F

Please circle answers as appropriate

Do you currently own a Smart-Phone? yes / no

If yes, what operating system does your phone use?

Android iOS (iPhone) BlackberryOS Other/Don't Know

How important would you class 'good' Mobile Phone signal **strength** in terms of your day-to-day phone usage?

Not Important Slightly Important Important Very Important

How important would you class 'good' Mobile Phone signal **speed** in terms of your day-to-day phone usage?

Not Important Slightly Important Important Very Important

Would you be interested in using a mobile application that can provide real-time details about your phone's received signal strength and speed?

Not Interested Slightly Interested Interested Very Interested

Interface Questionnaire

Please write answers as appropriate

Did you find the application easy to use?

Yes, very simple buttons

What feature(s) of the application did you like?

Speed test.

What feature(s) of the application did you think could be improved?

aesthetics of the app.

Would you consider this application 'useful' in your day-to-day life? Please explain your answer:

Not day to day but if I wanted to download something or watch something on the go.

Do you have any other comments or suggestions about the application?

Thank you for your participation