**TourInf, a Tourist Information Application for a Mobile Device on the Android Platform**

Final Report for CS39440 Major Project

*Author*: Adam Rigby ([ayr9@aber.ac.uk](mailto:ayr9@aber.ac.uk))

*Supervisor*: Rhys Parry ([rrp@aber.ac.uk](file:///C:\Users\Ad\Documents\Dissertation_Final_Report\rrp@aber.ac.uk))

14th March 2012

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a BSc degree in  
Computer Science (G401)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

**Declaration of originality**

In signing below, I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the sections on unfair practice in the Students’ Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
* I understand and agree to abide by the University’s regulations governing these issues.

Signature ……………………………………………. (Adam Rigby)

Date …………………………………………………

**Consent to share this work**

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature ……………………………………………. (Adam Rigby)

Date …………………………………………………

**Acknowledgements**

I am grateful to my mother Dawn Rigby for her continual support, guidance and backing throughout my University Degree, my uncle Mark Chetwood for his wisdom and inspiration, my grandparents for their motivation and great desire for me to succeed, my second parents James and Lisa Parton, whom without I would never have got so far, my best friends Dan Parton and Greg Norton, all my University friends including Adam Connah, Sebastian Graham, Andrei Stanica and Asa Carrington, and last but not least my wonderful girlfriend Sarah Evans.

I’d like to thank all the staff at Aberystwyth University, in particular Rhys Parry my final year supervisor who I wish a happy retirement, and the numerous lecturers who have helped me acquire a fantastic education from my years in Aberystwyth.

I would like to dedicate this piece of work to my late father Robert John Rigby who is so deeply missed but shall never be forgotten. He made me into the man I am today and I hope he would be proud of my achievements and this particular piece of work.

**Abstract**

This dissertation details and describes how I have created and developed a Tourist Information/Point of Interest application for a mobile device that uses the Android platform. For this major project I have created an application that mobile users and tourists can use quickly and easily to research destinations before travel and also use on the move within a town or city in order to find places of interest within that vicinity.

Once the user has found a place of interest using a category system they are then able to find out extra details about the area of interest, view their location on a map and get directions to their desired destination. The aim of the project was to achieve this functionality whilst still focusing on ease of use and speed of operation using a simple graphical user interface to achieve a user centred design. I feel I have achieved this objective with a project that offers the desired functionality whilst offering fine usability to its users.

**Contents**

1. Background & Objectives 6

1.1. Project Overview 6

1.2. About the App and Related Works 6

1.3. Reasons behind choosing the project 6

1.4. Project Goals 7

1.5. Related Works 8

2. Development Process 11

2.1. Introduction 11

2.2. Modifications 11

2.3. Platforms Considered for Development 11

2.4. Development Environment 12

2.5. GitHub 13

2.6. Object Aid Class Diagram Tool 13

2.7. Testdroid Automated Testing Recorder and Testdroid Cloud 14

3. Design 15

3.1. Overall Architecture 15

3.2. Initial Design 15

3.3. Android Application Concepts and Components 16

3.4. Detailed Design 18

3.4.1. Class/Activity Descriptions 18

3.5. User Interface Design 20

4. Implementation 23

5. Testing 30

5.1. Overall Approach to Testing 30

5.1.1. Hardware 30

5.1.2. Unit Tests 30

5.1.3. User Interface Testing 30

5.1.4. User Testing 30

6. Evaluation 31

Appendices 36

A. Third-Party Code and Libraries 36

B. Original Graphical User Interface Design 37

Annotated Bibliography 38

# Background & Objectives

## Project Overview

For my major project I have created a Tourist Information application to run on a mobile device that uses the Android operating system. The app (application/software) that I have developed makes use of several modern technologies to ensure it’s up to date with the current technology market. It has been designed with the present and future trends in mind, for use on smart phone and tablet computers and using a graphical user interface most suitable for the latest touch screen devices. The app makes great use of internet access made available to it through these devices to provide the necessary information. It uses the internet to send requests for data and receive the relevant information back, such as the tourist information data and any maps, directions etc. The app uses the Global Positioning System (GPS) to help find a users approximate location which then enables it to find tourist attractions nearby.

After being launched the first thing the app does is gain the GPS location of the user, once this has been obtained or after they have entered a destination manually, the app sends a request using the Google Places API [1] and the response to create listings of tourist information for any category that the user selects to find out more about e.g. Restaurants, Bars etc. Once these basic listings have been produced on screen the user is then able to select a listing which will present the user with more detailed information on the organisation or place in question e.g. Address and Phone number. The user is also given more advanced options on the detailed listing so that they can see the destination on a map or even get directions. For the mapping and direction elements of the project the Google Maps for Android API’s have been used in order to obtain the relevant information [2].

## About the App and Related Works

The app that I have created needed to be extremely easy to use for any type of user, for those who are very competent and also those that are new to using apps and advanced touch screen devices. The user interfaces of other applications on the market which offer similar features and services (nearby locations, details, maps) look difficult and confusing to use, such as the one found on the ‘Pocket Britain’ app. Whilst others are often dull and uninteresting to use. Consequently my app needed to have a very simple and intuitive GUI for people to use whilst still being attractive and appealing to the user.

After starting work on the project it became apparent that it was important that the app offered as broad a range of Tourist and general information as possible. To ensure the app needed to have an extensive amount of categories to select from. The result of this was creating to an ‘all in one’ application when it comes to Tourist Information that is genuinely useful and useable for all users. The problems found with existing apps such as the one listed above is that some target Tourist Information whilst others target the local area. It appears that no one has tried to target both audiences at once. The aim of the application is that it allows users to be able to download the application then be ready to use straight away, enabling users to find any place information they require within an area in a few virtual clicks and without having to learn how to use the application or type a large amount text to get the information they require.

## Reasons behind choosing the project

I find the topic of the project very interesting. The amount of tourist information centre visitors has decreased quite significantly in recent years [3]. Consumers are now using other sources of information to obtain the data they once found at these centres due to advances in technology allowing us to be able to access data at any time of the day, and now quite common to do this frequently and wirelessly through mobile devices over the internet, making it much more convenient for the user. This advancement has created a whole new digital market for this information which many companies are now competing in, in order to try and establish themselves as the market leader.

This project interests me greatly as it involves the use of mobile devices and the most up-to-date technology. I have always been interested in the latest technologies and have frequently saved up money to buy the latest gadgets and devices available in technology market. I am a huge fan of smart phones in particular. The idea of programming for such a device appeals to me greatly as it is an expanding technology and something that I and many others use extensively every day for an increasing number of activities. To complete the project I bought a tablet computer on the Android platform which I am finding increasingly useful to use in everyday life and I have thoroughly enjoyed developing the app for this kind of device as well.

I knew this major project would be very challenging to complete. I have never previously written an application for a mobile device. In order to complete this project I knew that I needed to learn a number of skills required from scratch. Writing software for a mobile device is quite dissimilar to projects that I have completed previously and I was aware that I needed to take extra things into consideration when designing and developing the application. I also knew that to complete the project I would have to learn how to use new software to develop the application. I was obvious that incorporating all the different features and technologies into one easy to use piece of software would be extremely challenging. Furthermore I knew an equally large challenge to writing the code would be creating a first-class user friendly interface as it is of great importance to the app and I did not have much experience of creating user interfaces, never mind for touch screen mobile devices.

As I saw the application as such a new and exciting challenge I knew that the project would be extremely worthwhile. The project would extend my knowledge in all aspects of Computer Science and what I have learnt throughout the course. The project would improve my programming skills and build on things I learnt from a human computer interaction module that I completed. Furthermore It gave me an opportunity to use the project management skills I have learned throughout the course and on an industrial placement to ensure that the project was completed successfully and on time.

## Project Goals

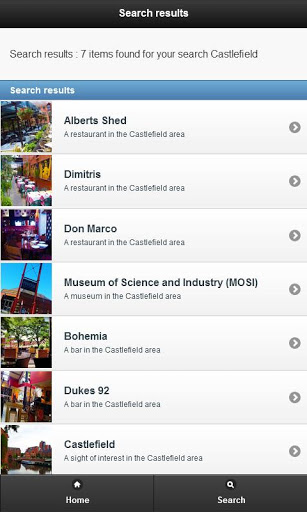
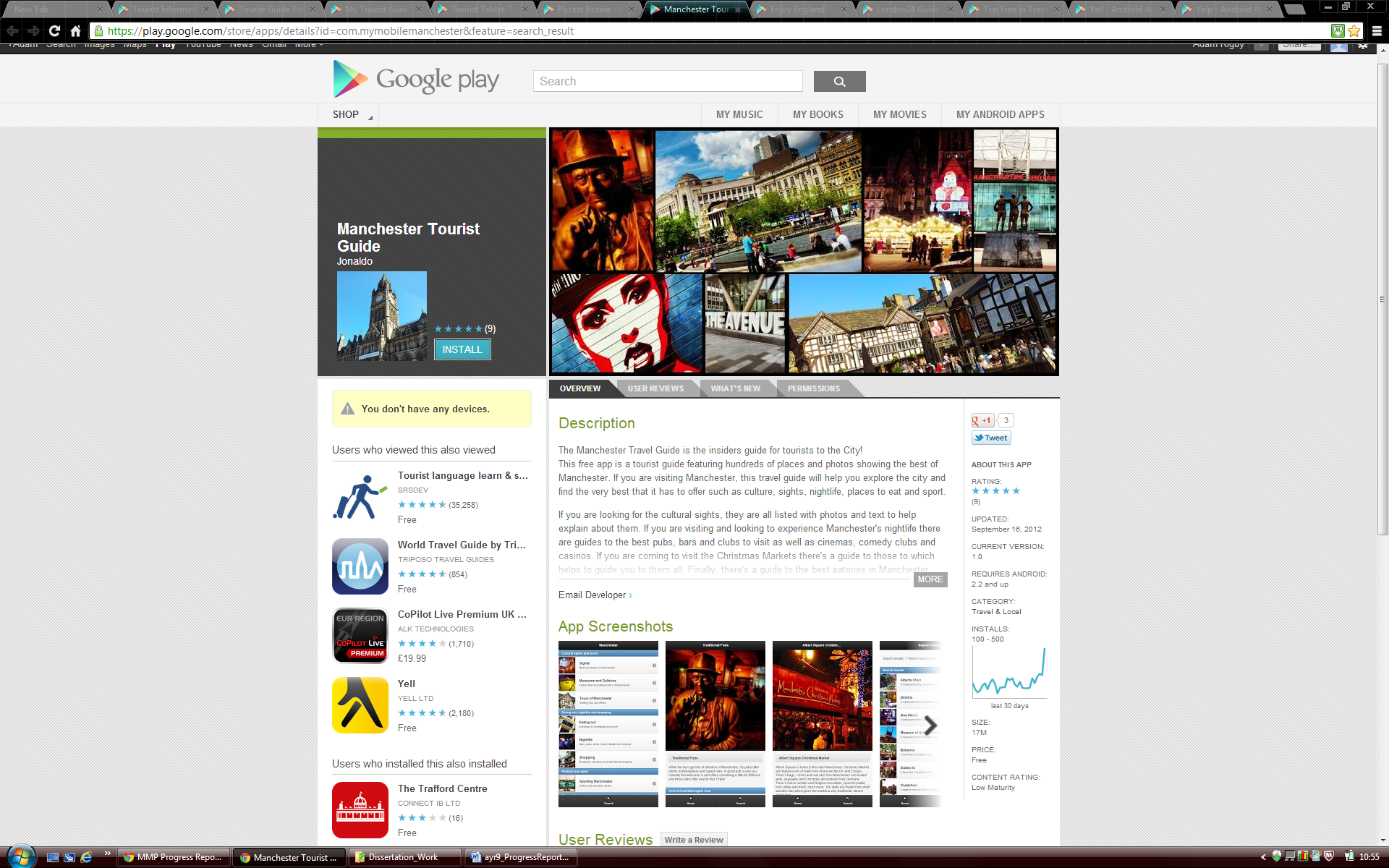
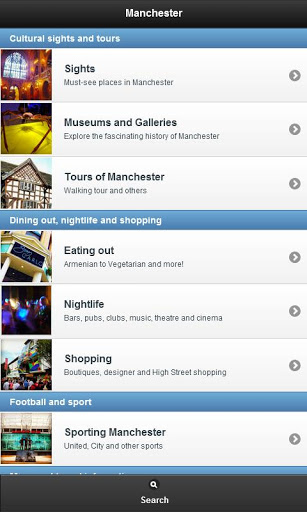
To evaluate the success of the project the application will be measured against the core goals set out for the project. The core goals are as follows:

* The app should be simple and easy to use for the user by having a simple icon based, touch screen focused user interface.
* To have an extensive list of categories for users to choose from and find information about.
* For it to produce accurate lists of data and information link to these categories
* To fully incorporate the more advanced features into the app such as Mapping and Directions whilst ensuring usability.

Desirable additions to these project goals were set which I would try to complete if the above goals were met and I had time to complete them. These would include the ability to display reviews of establishments listed. Furthermore the ability to add an image uploading feature to the app where a user would be able to take a photo of an establishment or a destination and upload it to the app so that other users could view these photos when they are going through listings would be a nice addition.

## Related Works

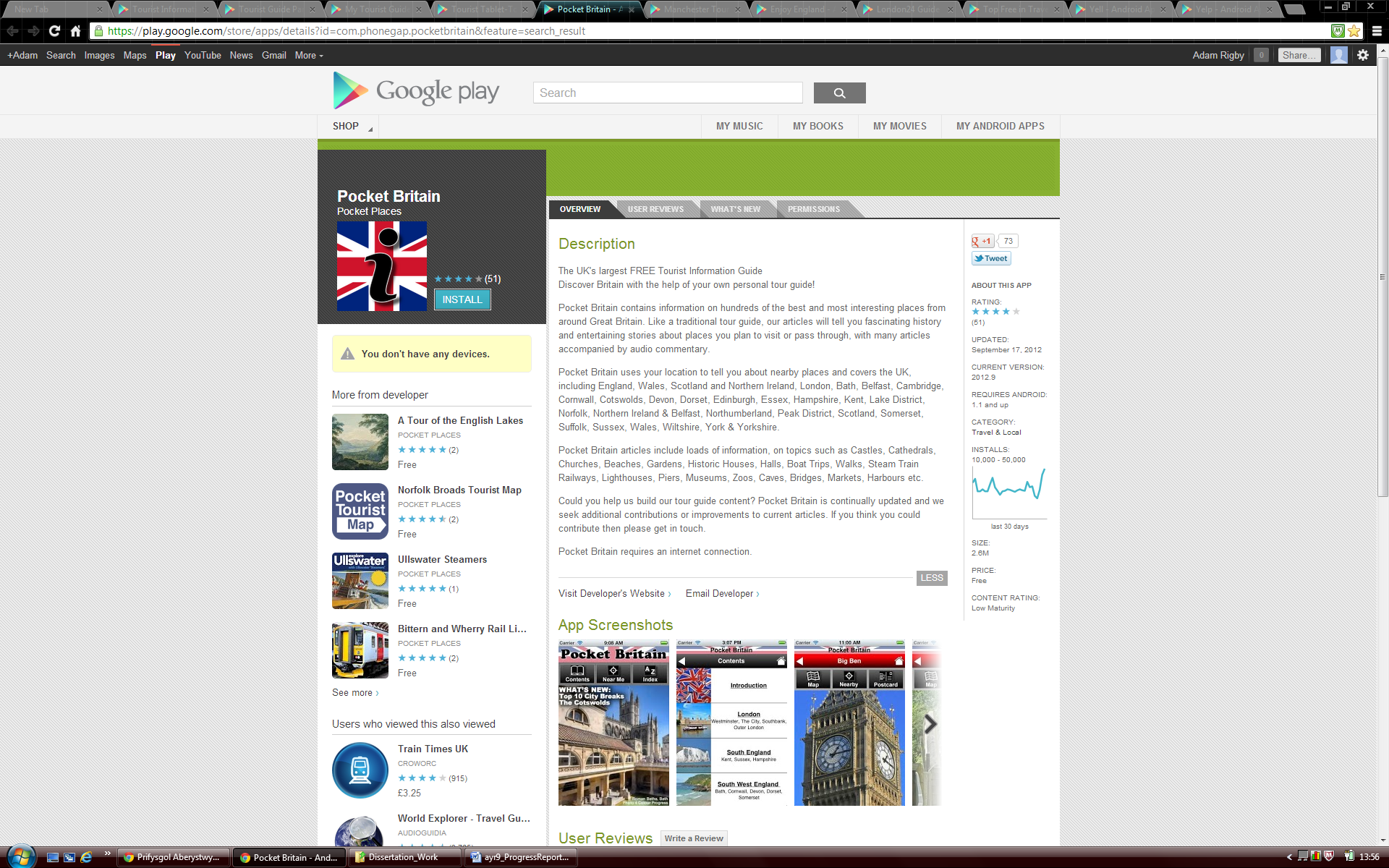
As must be expected with most software projects, there are similar competing applications already on the market, especially as the Google Play store holds over half a million applications [4]. Research was undertaken to look at similar applications on the market in detail, studying about them in the Google Play store and downloading a select few to look at in more detail. An example of a tourist app for a singular city is the Manchester Tourist Guide by Jonaldo[5]:



After downloading and looking at this application it became apparent that it is quite a nice app for people to use. It uses a different graphical user interface design to what has been developed for this project in the tourist information app. The application uses what is known as a list view in android development which has been modified to include pictures. This is a good way to display information however it can take up a lot of screen space when compared to using the grid view that I have used on my application. For the initial screen that appears, it was therefore decided that a list view shouldn’t be used as it is likely to take a user longer to find the information they want. This is because they may need to read more information and scroll further down the screen to obtain information when compared to something like an icon based application.

On the other hand for other parts of the application where more textual information needs to be displayed, it is appropriate to use a list view. Another plus point about this particular application is the depth of information it has on destinations and establishments. For each place listed it has a large description and a detailed photo. This was something that would have been nice to replicate in the application, however, unfortunately a web service with this detailed information could not be found and used in the project time period making it not possible to reproduce. As the application aims to cover the whole of the UK It would not be possible to describe and write about thousands of major locations. This app has some great options such as ‘directions’, ‘view website’ and to call a particular listing. These are all features that were decided that they should be included in this project.

Pocket Britain [6] is another application that has been researched. Unlike the Manchester Tourist Guide this particular application does provide information for the whole of Great Britain. It is also dissimilar to the application above in terms of appearance.



From when this application was first downloaded the Pocket Britain app appeared to look rather unprofessional and not well presented. First impressions can quite often count and after evaluation the thought was that users are not likely to want to use this application because of its appearance and poorly laid out graphical user interface. Due to the user interface it took quite a long time to find the destination or specific place you are looking to find.

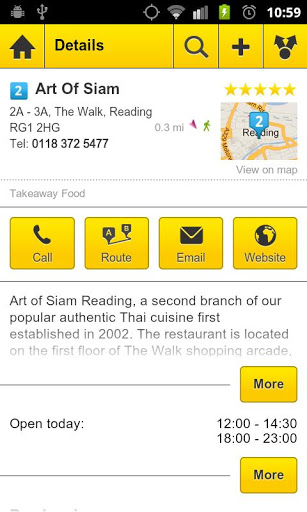
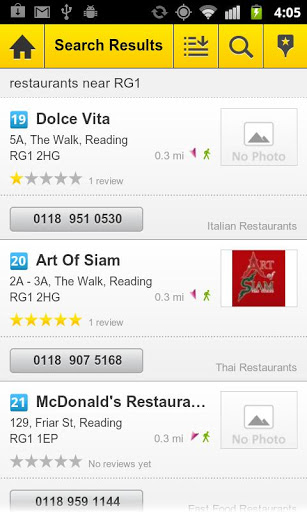
However the content of information is fantastic. This will appeal to some users who want in depth knowledge of an area but not others who just want an overview of their surroundings. The app also seems to concentrate on historical buildings and museums. Whilst this is not a bad thing it is likely that most people will also be interested in finding amenities and other attractions in the local area.

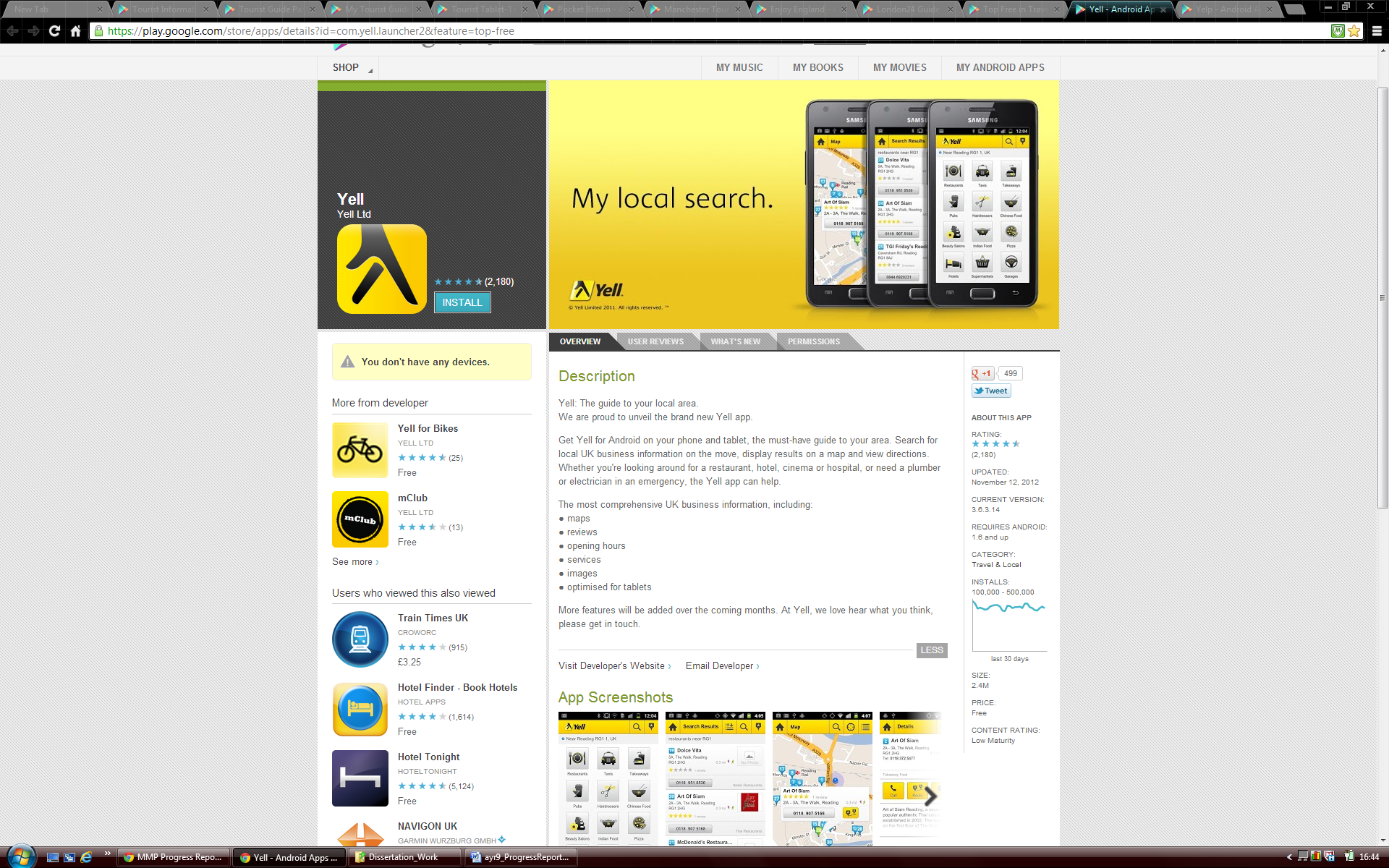
This particular app also has a ‘nearby’ feature, which lists other towns and cities in the local area to where you are or a place that you have selected, you can then look at the tourist attractions linked to that place. This is a good feature and it would have been nice to include a similar feature in my application. The app also offers a user a map of the listing they wish to see with street view enabled which is a nice touch, although unfortunately directions to the destination from your existing location is not supported.

The two applications that have been reviewed and assessed above were found from doing a search for ‘Tourist Information’ in the Google Play Store. However this research has shown that the app created is more of a hybrid application mixing aspects of these apps with those found in local area guides and business listing apps. After visiting the ‘Top Free in Travel & Local’ section of the Google Play Store the Yell application was discovered [7].

These types of applications seem closer in style to what was envisaged with regards to this project. The Yell app uses a simple icon based structure in its main menu, very similar to the one that has been implemented in the application for this project and produces listings and listing details in a well organised and easy to use manner. However there are a few minor drawbacks with the application. One is the icons on the home screen are very large and with only three per row which means you have to scroll down to find categories despite using an icon based interface. Another is that perhaps the application could be called plain with basic icons and minimal colour used in the application, although on the other hand this is also an advantage as it makes everything easy to read and understand.

When downloading the app it was noticed that the tablet version of the application is drastically different to the smart phone version and doesn’t function anywhere near as well, with usability being a major issue. It is likely that it would be very confusing for users to operate the application if they have used one version then attempt to use the other on another device.





The need for consistency across devices became apparent after seeing this and needed to be taken into account when building and implementing the Tourist Information app for the project. The app needs to cater for both phones and tablets and should try and compensate for all screen sizes. Unfortunately on Android the information displayed on screen does not automatically shrink to suit the device it’s being displayed on. This needed to be taken into consideration when developing the application for the project.

To conclude, a great deal was learnt from the research completed on other applications available for mobile devices. It helped greatly when trying to understand the requirements and aims for the application that needed to be built. It also highlighted that the appearance, usability, user interface and compatibility are the key elements when developing a mobile app and ensured that these findings were taken into consideration throughout the project.

# Development Process

## Introduction

Originally the Waterfall Model was chosen to be used as a methodology forth this project. This model was chosen as it was felt that it best meets the requirements of the project. Having learnt about Agile Methodologies in a module this year within University I would have liked to have used this method as it is something that is new to me and I agree with most of its principles. However after analysis it became clear that I wouldn’t be able to practice enough of its core components to make good use of the methodology and use features of it such as having customer collaboration and pair programming in the model to make it work effectively.

The major project that has been completed when compared to software projects in industry is relatively small and it relies upon just one person as an individual for it to be completed and not a team of developers. For this reason the Waterfall model is therefore the most appropriate model to use for the project as it keeps things relatively simple and gives a disciplined process. Often other methodologies such as Agile and Spiral are more suited to larger tasks and being used in a multiple team environments where projects are usually more complex. The requirements for this project have been stable from the beginning and throughout meaning that the flexibility and ability to change that comes from using Agile Methodologies wasn’t particularly needed.

Due to the Waterfall model being chosen for the project, the planning that was completed relates to this and takes the relevant stages into consideration ([please see appendix 5.1](#_Dissertation_Project_Plan)). To demonstrate the week by week planning of the project graphically a Gantt chart was created with milestones for project deliverables ([please see appendix 5.2](#_Project_Gantt_Chart)). As would be expected from using the waterfall model a requirements specification, design specification and test specification were included in the Gantt chart to be delivered as part of this final project. These steps needed to be completed in the right order and on time to follow the methodology correctly.

## Modifications

Did you have to modify the model to suit a one-person project? If so, what did you change and why?

Once the project began a decision was made to modify the planned choice of methodology from the waterfall method to a more prototype based model. This decision was taken due to the demands of the project that were only realised once project research had started. Having no experience of developing a mobile application for Android before meant that being able to produce a reasonably accurate design and test specification before starting development would not really have been particularly viable.

## Platforms Considered for Development

When you have decided that you wish to create an application for a Mobile Device you have a number of different operating systems you can choose to develop for. Currently the most used operating systems on Mobile Devices that are also the most popular with mobile application development are iOS, Android, and Windows Phone. The iOS and Android operating systems are however the two dominant forces in the mobile device operating system market [16]. Due to their large share of the market (72.4% of the mobile device market use Android and 13.9% iOS) it was decided that only developing an app for one of these two operating systems would be considered and evaluated. This is because the app that was to be written needed to have the largest possibility of being used and exposed by the largest audience possible if at a later date it could be considered for release on an app store and sold. By releasing an app on one of the two largest platforms it is more likely to gain popularity and gain more money from any fees and advertisements. As a developer you do not want to spend a large amount of time producing an app for an operating system that doesn’t have a following and has a small amount of users.

Ultimately it was decided that the application would be developed for the Android platform. This decision was made for a number of reasons. Firstly the minimum hardware requirement for iOS development is having a Mac with an Intel processor. In addition iOS apps can only be developed on the Mac OS x operating system [17]. There are different ways in which you can navigate around these problems such as creating a ‘Hackintosh’ to run Mac OS X on a non-apple PC, using an emulator or running the operating system in a virtual machine [18]. However if it is avoidable you do not want to do this as these methods are in most cases illegal and can often have many problems in operating correctly. As a windows user it wasn’t viable to spend a large amount of money buying a new computer specifically to complete this project.

The other major problem with writing an app for iOS was that for iOS 6 apple decided to drop using Google Maps and start using their own mapping solution. It has been greatly publicised in the news how Apple’s new maps app launch has been very unsuccessful with the maps not working correctly and with features and destinations missing [19]. After the issues had been discovered users of Apple products weren’t sure if Apple would continue to use the new maps service or revert back to using Google Maps. At this moment in time it appears Apple are determined to carry on offering their own service although Google have now created a map app for Apple devices that is on the Apple App Store . With these mapping problems arising at the time of starting this project and with the need to use a stable mapping service for features of my Tourist Information app, it was more logical to develop for the Android platform. Android is owned by Google and therefore the platform is very unlikely to move away from using its owners mapping services and API’s, this decision resulted in reducing the risk of creating the project unsuccessfully.

My belief is that Android was by far the best operating system that I could developed an app for on a Mobile Device. It is currently the most popular and used operating system on mobile devices [16], it allowed me to develop on hardware I already own, and most importantly it has the correct services and development API’s that have allowed me to create a good Tourist Information application. What’s more when writing for Android the programming language used is Java. Whilst learning new programming languages is very beneficial and interesting, extending my knowledge and skills in the Java language that I have used throughout my time at University will also be extremely useful.

## Development Environment

To start writing software for Android mobile devices it is a good idea to work in an integrated development environment (IDE) due to the advanced structure of such applications. This is because Android applications are constructed from much more than just Java source files. Before development started research was undertaken into finding tools and environments you can use to help create an application for the Android platform. The options that were discovered and evaluated were the NetBeans IDE used with a plugin that can be found on the Kenai web site [11], the IntelliJ IDEA IDE [12] or finally the Eclipse IDE with the Android Development Tools (ADT) plugin [13].

After evaluation the Eclipse environment was chosen to be used for the project for a number of reasons. One such reason is that previously I had mostly used Eclipse for development projects so I am accustomed and familiar with using the main parts of the program. I have only used Net Beans briefly for one assignment and came to the conclusion that I prefer the Eclipse IDE due to its layout and ease of use. Whereas I was not aware that the IntelliJ IDEA IDE existed as I don’t believe it is commonly used and I would prefer to use a tried and tested product. However the most important reason behind this choice was the fact that the development section of the official Android developers’ site highly recommends using Eclipse a long with the ADT plugin. The site also helps you install and setup the appropriate environment with some useful tutorials []. From browsing the web researching this topic also became clear that this IDE was the most popular with Android developers assuring that it was a good choice. In order to meet the software requirements for the program the newest version of Eclipse had to be installed. Once this was completed the Android SDK and ADT plugin for Eclipse could then be downloaded and installed. After this had been setup and configured the development work for the application and project could start.

To start developing and getting a feel for Android development the ‘Building Your First app’ was the first tutorial that was followed and completed. This can be found in the training section of the Android developers’ site [14]. This was a very trivial exercise, and it basically involved creating a new ‘Hello World’ project. However it did introduce me to the Android toolkit, the structure and layout of what Android apps should look like, and how they start to be created. To build on this initial knowledge I attended several of the Computer Science department’s Android programming sessions at University [15]. From these sessions I learnt a lot more about the make-up of an Android application and built and modified some basic Android programs. Those worksheets were really useful in extending my initial knowledge of writing software for Android and working on these worksheets definitely helped when coding of the final project started as I had a good basic understanding of how to develop for the android platform.

## GitHub

GitHub is an online project hosting service that is often used for software development projects that use the Git revision control system. GitHub is a well used source control system in this field and has many great features. GitHub was used on this project for a number of reasons. One major reason is that the service that they offer is very easy to setup and use as your repository is so easily accessed online. GitHub also offers a GitHub for Windows application client. This is a great piece of software and was very useful as a project tool as you can simply drag and drop files into the application which then allows you to quickly commit and synchronise your work and any changes you have made before sharing the files online. The GitHub service and Windows application are also available for free as long as the code is made public and open source which I do not object to.

## Object Aid Class Diagram Tool

Complicated diagrams such as a class diagram for an application such as the one in this project can be difficult to create and also very time consuming. Often if these diagrams are created manually they are not well presented and can be confusing to look at. For this reason tools to automatically generate the diagram from source code were researched. The object aid tool was used as it seemed the most appropriate as it offered the correct functionality and was easily downloaded and installed into eclipse as a plugin**.**

## Testdroid Automated Testing Recorder and Testdroid Cloud

In order to test the application thoroughly it was decided that on top of the usual testing methods an automatic user interface testing tool could be used. Research was completed on the Internet about what tools were available to do this and several tools were found. Some of the tools found were...

# Design

## Overall Architecture

After deciding a set of project objectives, aims and goals and creating a list of functional requirements, the design for the system could start to be considered. What became apparent was that there were essentially four major building blocks that needed to be built and integrated together in order to create a full working system with the desired output. One of these major building blocks for the application is the user interface which is inherently important for this application for usability and ease of use. It has to let users navigate between the different screens of the app simplistically and allow easy access to its functionality. Another one of these blocks is the part of the application that uses the Google Places API as a web service to request and receive the place data information that is to be displayed within the application.

The mapping elements of the software is again a major section of the application that is used to display and mark the locations of the places returned from the web service on an interactive and fully detailed map. This component of the app will also let the user obtain directions to any of the places that they have chosen as a destination from their current location. Finally the last major part of the application is the modules that take care of the GPS and Internet connection. This part of the application is extremely important as without these classes performing correctly the rest of the application would not be able to function either at all or as required due to the other parts of the application relying on the use of these classes to operate.

## Initial Design

In order to start creating a design for the application to be developed, a use case diagram was made. The use case diagram that can be seen below in figure x shows all the actions that the user can perform in the application.

Show all places on a Map

Show place on map

User

Get Directions

View Website

Use Current Location

Enter Location

Select Category

Place List

Call Place

Fig. X

After creating this simple diagram it became much clearer as to what work needed to be produced for the application and gave a good basis to start thinking about the design in a bit more detail. To understand how these actions could be incorporated in the application and how they would perhaps link together and operate another diagram was created. This further helped the understanding of the project and how it should be designed. Figure X below shows this diagram, it holds a resemblance to a flow diagram and details the application flow from the home screen to the more detailed features using some of the actions listed in the use case diagram above.

Show on map - Action

Get Directions - Action

Go to website - Action

Call - Action

Home Screen - View

Main Menu - View

Use Current Location - Action

Enter Location - Action

Select Category - Action

Near Places - View

Select Place - Action

Show all place on a map - Action

Map (All) - View

Place Details - View

Map (Singular) - View

Map App - View

Browser - View

Phone call - View

Fig. X

Once this diagram was established it then allowed the design to continue developing, getting more detailed. After getting this far with the project and doing the research stated in section 1 of this report it became apparent that to create a design more appropriate to the application a different development process would be needed as stated in section 2.2. The diagram was also a great help in designing the user interface and how it would function as discussed later in section 3.4.

At each level of the diagram, items are labelled as either a ‘view’ or an ‘action’ where a view needs to be a screen displayed by the application and ‘actions’ are the things that need to be performed in order to navigate between the different views/user interface screens. From this we then have the basic structure of the application and one that can now be designed in more detail.

## Android Application Concepts and Components

There are a number of terms that are given to various components of Android applications that are very important in development. For the project these components needed to be learnt about and understood for firstly the design and then the implementation of the application.

The main components of Android applications are called Activities. ‘An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email application, each one is independent of the others. As such, a different application can start any one of these activities (if the email application allows it). For example, a camera application can start an activity in the email application that composes new mail that can then allow the user to share a picture.

An activity is implemented as a subclass of Activity in a java source code class’ [REF]. The activity holds all the source code for a single screen and also obtains and uses an associated XML layout file to produce the user interface with the correct functionality.

Therefore when we look at the diagram above in figure X everything that has been labelled as a view needs to have its own activity, very much similar to having a java class apart from it having additional properties that can be utilised to aid user interface communication and construction. To use components such as activities you often need to activate them by sending an asynchronous message called an intent.

‘Intents bind individual components to each other at runtime (you can think of them as the messengers that request an action from other components), whether the component belongs to your application or another. An intent is created with an Intent object, which defines a message to activate either a specific component or a specific type of component. For activities and services, an intent defines the action to perform (for example, to "view" or "send" something) and may specify the URI of the data to act on (among other things that the component being started might need to know)’. For example, an intent might convey a request for an activity to show an image or to open a web page. In some cases, you can start an activity to receive a result, in which case, the activity also returns the result in an Intent (for example, you can issue an intent to let the user pick a personal contact and have it returned to you—the return intent includes a URI pointing to the chosen contact)[REF].

Again when we look at figure X the items that are labelled as ‘actions’ are in most cases the intents that are needed to activate the use of other components. An Android application must also have a manifest file present that is written in XML. This file must declare all components e.g. activities that are in an application and should also declare all application requirements for example the minimum version of Android required and any hardware configurations and permissions required [REF].

Throughout Android projects non-code application resources (images, strings, layout files, etc.) are used via reference. In the tree structure of an Android project is a directory named res (resources) where all these resources are stored in various folders. The layout files for an application are written in XML and are used by activities to display the correct user interface. These are held in the layout directory. From this we can understand that application built on the Android platform use an MVC framework. For example the XML layout files represent a view, the activities are the controllers and the model is represented by services and data held within the application (locally stored database or data retrieved from external sources). Therefore the project is using the MVC architectural pattern.

Strings that are not declared programmatically in the application e.g. in the layout files to be displayed are declared in a String.XML file which is within the values directory. Lastly images and other items to be displayed are stored in the various forms of drawable folders held in the res directory. There are several of these folders as each folder is used for different screen resolutions. From learning and understanding all of these key Android concepts and how these types of applications are structured, the design could then be developed further.

## Detailed Design

In order to continue detailing and creating a more in depth design of the system a class diagram was created to establish what activities/classes were going to be needed to create the desired application with the required functionality. The class diagram for the application can be seen in appendix x where all the activities that are required are shown with the relationships they have between each other along with the methods and variables that they have.

As the application has been written in Java the project has been developed with an object orientated approach. As briefly mentioned earlier, each view of the application outlined above will have its own activity/class with additional classes being used to support these activities.

### Class/Activity Descriptions

**AlertDialogManager Class**

This class is used to display dialog boxes at various stages of the application when needed. They are usually used to display a message or warning when something is not present or not allowed e.g. if no places of the type selected can be found in the local area.

**AllPlacesMapActivity**

This activity creates a map using the new Google Maps for Android V2 API. The activity gets the complete list of places for a category that has been selected and sets a marker on the map where each of the places is located. Each marker is clickable so that the address of the location can be seen. The location of the user can also be added to the map. To make sure the map is centred before being zoomed in, the average position of the markers is calculated.

**ConnectionDetector Class**

This class uses a system service in order for the application to be able to make a connection to the Internet. The state of the network is also checked to ensure that the connection has been successful and can be used by the application.

**GooglePlaces Class**

The GooglePlaces class connects to the Google Places web service and makes a request for places depending on the parameters that its methods are given. The class then parses the results that it receives into the serializable classes of the application so that the information received can then be accessed and used by other classes of the application.

**GPSTracker**

The GPSTracker is a service component of the application. The service class implements Androids LocationListener class to receive notifications from the LocationManager when the location has changed. The service class checks what GPS and network providers are available for usage and then will use the most appropriate provider (GPS if enabled) to request location updates from them depending on parameters to keep and update location of where the device is. Methods to get current latitude and longitude values are provided so again they can be used in other parts of the application.

**ImageAdapter Class**

The main menu of the Tourinf application uses what is called a grid layout and is defined in its associated layout file. However for this to display the images and text correctly in the grid a customer adapter needed to be written. The class uses an additional layout file to put the image and associated text into it, which is then put inside the grid view, another layout file. It has been optimised so that the main menu that it helps create can be displayed correctly on multiple screen sizes.

**MainActivity**

The MainActivity class essentially creates the application home screen that you see when the application launches. It forwards the user on to the main menu a long with any location that has been entered manually by the user for the application to use.

**MainMenu Activity**

As suggested this activity displays the main menu to the user by making use of the ImageAdapter class described above. The activity contains a HashMap of key value pairs where an int grid position of the image is associated to a Google place type. Once an icon has been clicked this place type is forwarded to the MainPlacesActivity class for it to make a places request. If a manual location was entered previously, this is again forwarded to the same class for the request.

**MainPlacesActivity**

The main places activity is one of the major components of the application. It makes calls to the ConnectionDetector, GPSTracker, and GooglePlaces classes in order to produce a list of places for the type selected by the user. The list is displayed on screen and when one of the places is selected the reference to get the further details of that particular place is forwarded on in an intent to the SinglePlaceActivity.

**MapActivity**

Similarly to the AllPlacesMapActivity this activity creates a map using the new Google Maps for Android V2 API but only for a singular place, having only one marker present.

**Place class**

The place class is a serializable class in which the key JSON values obtained through the Google Places web service request are stored. All the keys can then be accessed and used inside other activities. It can be observed how the majority of these values are used in the SinglePlaceActivity.

**PlaceDetails Class**

This is another serializable class that holds the status of the JSON which has been parsed and its complete result (its details) for a single place/location in the form of a Place object.

**PlacesList Class**

The last serializable class used in the application is the PlacesList class that again holds a status for JSON that is attempting to be parsed and also its results for one or more places within an area, again as an array of Place objects

**SinglePlacesActivity**

The SinglePlacesActivity makes a call to the Google Places class to getPlaceDetails in order to get more details about one of the places that the user has selected from the list. The activity displays the key details for a place e.g. name, address, rating that are stored in the Place serializable class. If an image of the Place is referenced in the JSON returned from the web service then this will also be displayed in the graphical user interface. The activity also hosts the more advanced features such as Show on Map, Get Directions, View Website and Call a Place. These are displayed to the user with a button which when clicked will perform one of those actions. To show the place on a map, the MapActivity gets called, whilst for the other three features, intents are sent. Some buttons for these features are set to display as invisible if they are not present, for example the website and call button will only be displayed if the place in question has a phone number or website URL in its place results.

## User Interface Design

As this project is about creating an Android application the graphical user interface design is of great importance due to the nature of the platform and the fact that virtually all Android application are displayed graphically. Two of the core project goals for the application were to ensure that:

* The app should be simple and easy to use for the user by having a simple icon based, touch screen focused, user interface.
* To have an extensive list of categories for users to choose from and find information about

In order to achieve both of these goals the design of the user interface needed to be of a high standard, taking the above points into consideration. When users download an app, they expect and want an application that will be nice and easy to use and that will not be complex and confusing to operate. They want it to be simple and for it to function and perform as expected, giving them the output that they sought after.

As has been detailed earlier in this report, the goal of the project was to use the latest technologies and think about using the most up to date techniques. Because of this it was decided that the application would be designed predominantly with the use of a touch screen in mind when used on a mobile device, whether it be a tablet computer or a mobile phone. This was decided as it appears that all tablet computers available on the market use a touch screen and the large majority of phones and smart phones now have touch screens, especially those that can actually download and run applications from the Google Play Store.

To create a good graphical user interface the design of it should not only be simple and appropriate for the device in use but also appealing and aesthetically pleasing to the user. All these aspects were kept in mind when designing the graphical user interface. Before the implementation of the project some mock ups of what the graphical user interface may look like were created and can be seen in Appendix B. Some minor changes have been made for reasons identified during the implementation and these will be detailed in the description of the different screens/views below.

The home screen for the application has been designed to be very simple as it doesn’t need to offer the user much functionality, the home screen only gives the user two options, to either use their current location in the application, finding places within their immediate surroundings, or to enter a location in order for the application to find places surrounding the town/city name or postcode entered. The home screen can be seen below in figure X.

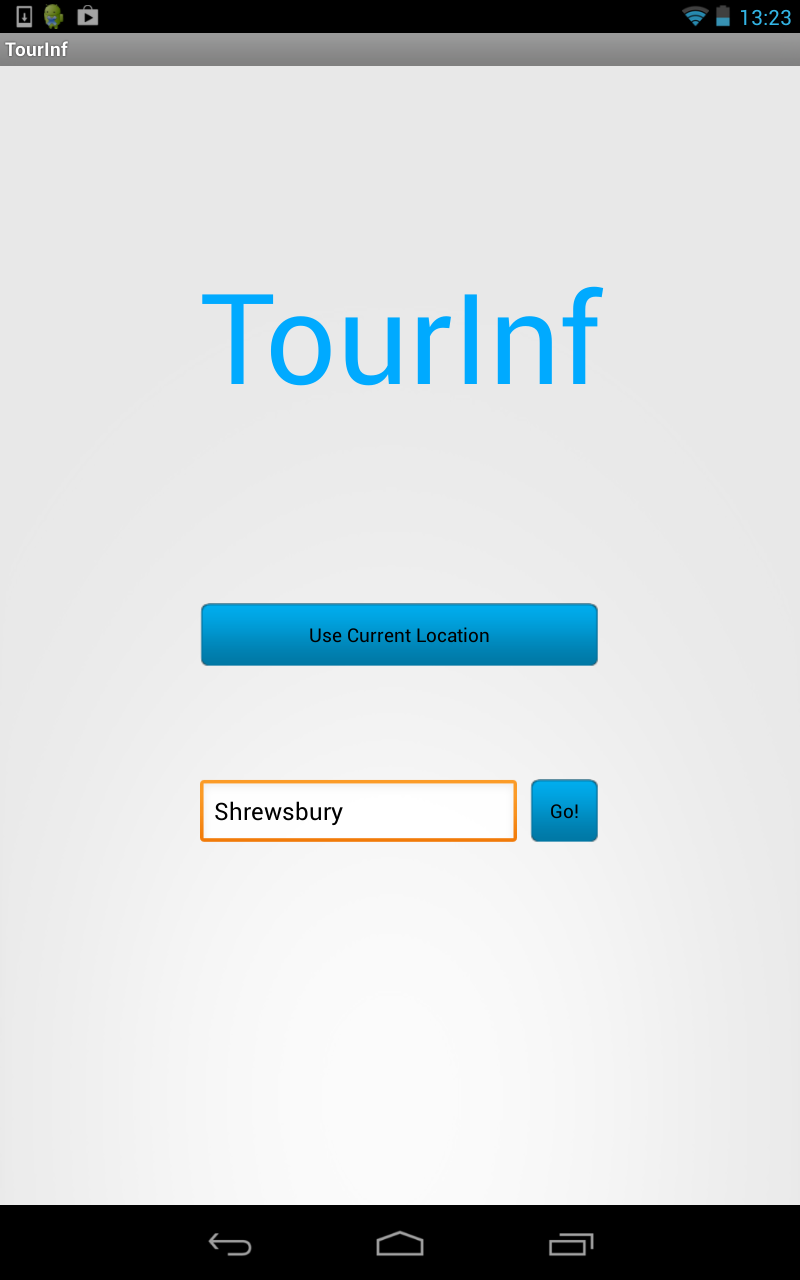


Fig X. The application home screen

The design of the next screen that the user is forwarded to was very important as it is the main menu for the application. It displays all the categories of places that can be selected to see if any places are in the surrounding area for that category. For this screen to be displayed an icon based design was chosen as it gives the user a quick graphical representation of the different categories, allowing them to quickly identify what they are looking for and allow for quick selection. Another reason this type of layout was used is that although it uses graphics it also uses screen space efficiently allowing a lot of categories to be displayed to the user at once. This enhances the usability of the app due to it being simple to operate and quick to use. Furthermore an advantage of using this layout is that a lot of operating systems such as Android and iOS used on tablets and smart phones use this type of interface to launch applications from their home screens. This should give the user a sense of familiarity when using the application and allow the user to easily navigate to the category that they want to select.

Finding appropriate icons for the design of the application was a difficult task. At first different images were from Google searches were used and tested. However this creation did not create the vision of a professional application. The images were all of different styles, had to be cropped to size and contained different colours. It was then decided that a set/pack of icon images was required to have consistency across the icons and giving the professional look and feel that was wanted. These icon packs, especially good ones are notoriously difficult to find, especially ones that are available for free and without licensing. After a lot of research and searching an open source project set of icons was found [REF]. These look far more professional and most importantly were offered in vector format. This allowed them to be easily resized appropriately using Adobe Illustrator of which a friends copy was used. Not all icons were present so a couple had to be created from scratch, but thankfully they weren’t the more complicated icons needed. The main menu from the application can be seen below in figure X:

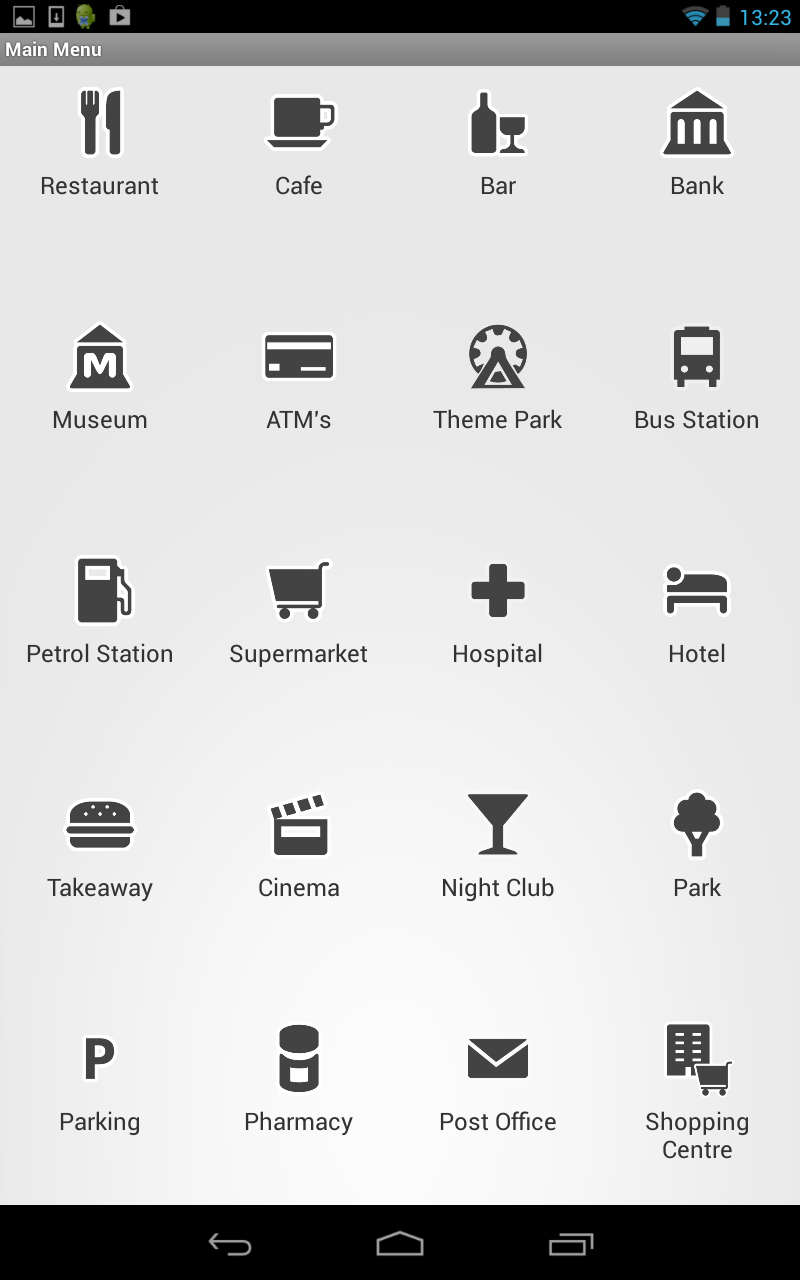


Fig. X. The apps main menu

Once the user has selected a category in the main menu, a list of the places related to the category chosen then needed to be displayed. After analysing and evaluating the different methods to do this, the often used basic version of Android ‘list view’ was selected to be used by the layout file for this activity as it is a clear and effective way to display lists of data. This decision is actually in contrast to what was drawn up in the initial mock ups of the graphical user interface (appendix x). In the original design an expandable version of list view was used. However this method of displaying the data was unnecessarily complicated. It is very difficult to create a well laid out user interface using this view. It also offers the users less space to view the places associated details. Additionally it may also be harder for the user to understand and select than having the more detailed information on a separate screen page. So for this reason instead of having one page that displayed the place name and expanded to show place details, these features got spread out between two different screens with the details and more advanced features being displayed on another screen separate from the list. The original anticipated design can be seen in appendix X (screen mock ups 4 and 5) whilst the finished application user interface design can be seen here in figure X:

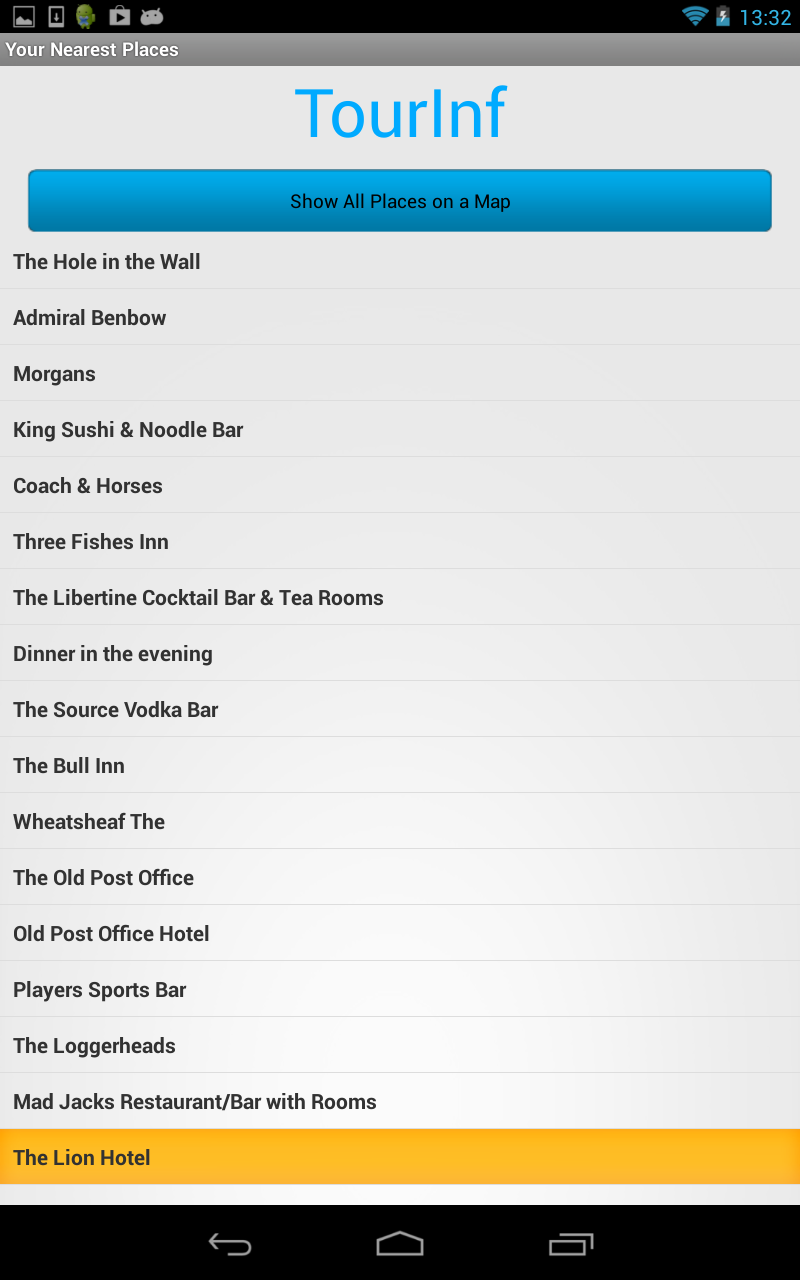
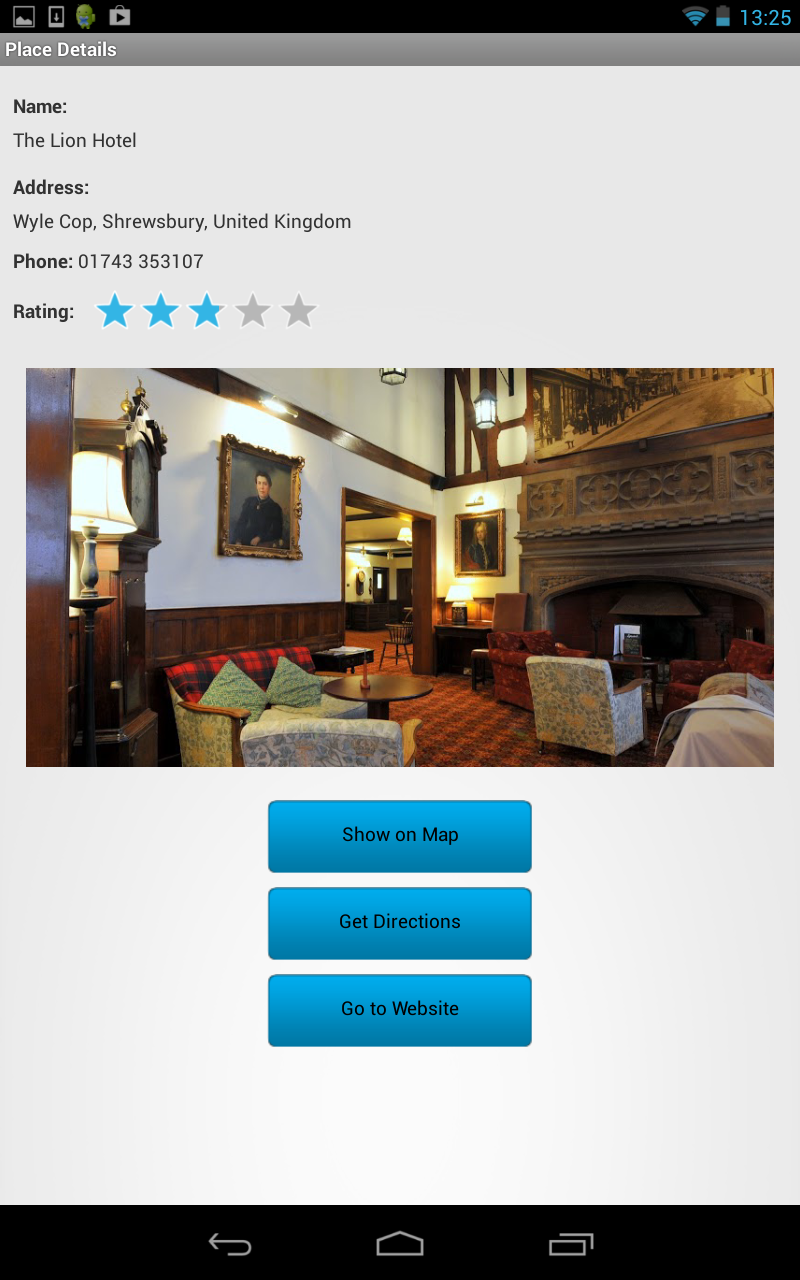


Fig. X. The list of places for a type which when selected goes to the details screen.

The list view that has been used in the screen shot above is a view group that allows a user to see a list of separated and scrollable items from which they can then click and select the item that they want. This layout was selected due to it offering simplicity to the user and the design. Similarly to the main menu screen, by using this type of layout in the view more categories could be seen at once and users would not be puzzled with the layout. Furthermore it allows the application to easily move on to the next screen of more detailed information as each item is selectable. Once clicked it shows the place details information screen.

At the top of the screen/activity is also a button that allows the user to see all the locations of the items displayed below in the list on a map at once. This then allows the user to see where the locations of the listed places are with retrospect to their own location. This is an additional feature that was not expected to be included and therefore was not included the original screen mock ups. The button to access this feature was put at the top of the screen so it is immediately seen when the screen is displayed and has been coloured in blue to match the house style of the brand that has been created for the application. The button was not but at the bottom of the list as users may not have realised the feature existed. An example of the map that may be created by this feature in the user interface is shown here in figure x:

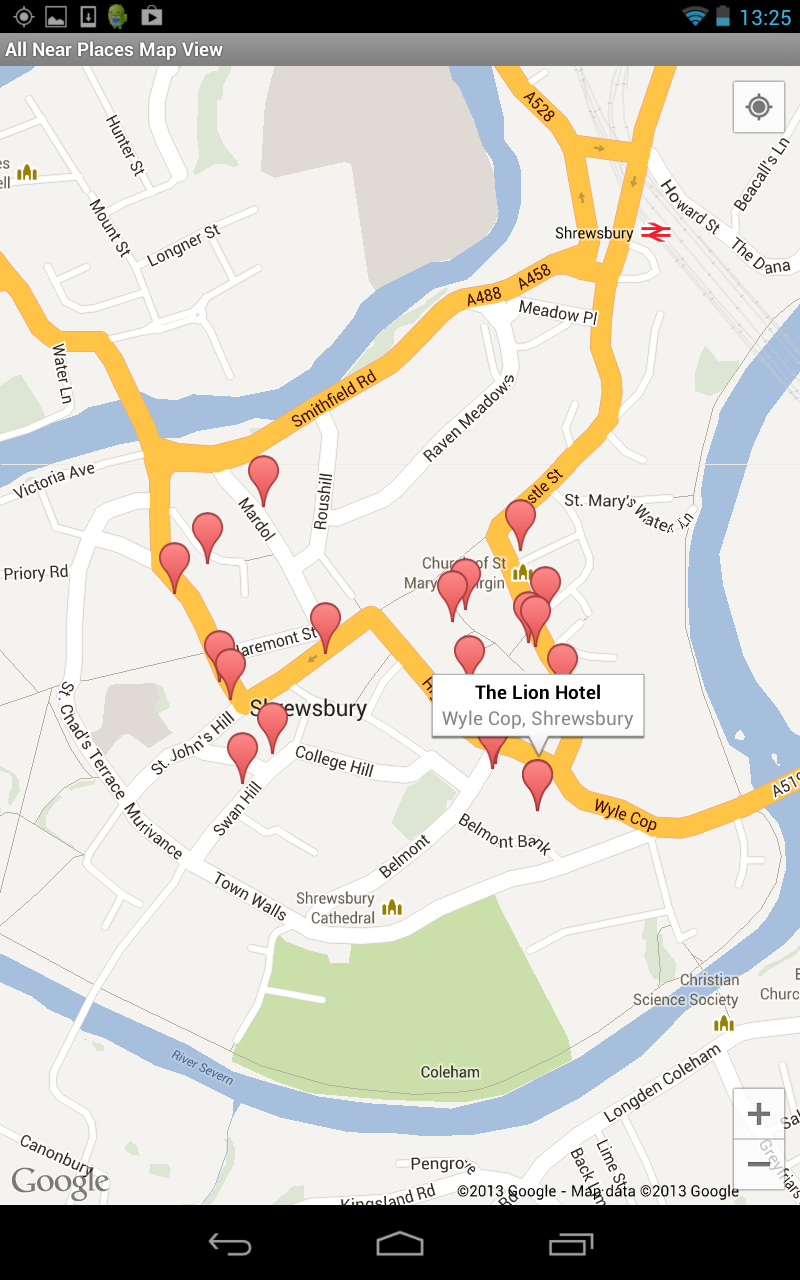


Fig. X. The all places map view.

The last screen that the user of the application can navigate to with the most detailed place information is the place details screen. This has a fairly simple design because it is the part of the app that needs to display more textual data and therefore has to be well organised and easy to read for the user so that they can understand and use the information they are being given. The place details activity uses a relative layout to ensure that everything can be positioned in an ordered and structured manner and where items are placed according to their siblings that are also in the layout to maintain their structure.

Several additional features have been added to the application when compared to the original design and mock ups and these have been included and displayed in the user interface. An image of a place selected is displayed on this screen if a reference is found to one within the results of the Google Places request. In addition a rating bar has been created to show the rating that has been given from the review given on Google. Several buttons that offer the more advanced feature of the app (show on map, get directions, view website) are clearly placed below the textual information and are again coloured in blue to create consistency with the house style of the application and highlight that the item is a button that will launch another feature. The ‘call’ button has also been added to the screen to use the additional feature that has been added. Although it must be remembered that buttons such as ‘view website’ and ‘call’ will only be displayed if the place in question has the website to visit or phone number to call. This is also true for the rating bar, if no rating is present, simply not stars will be coloured in. An example of the details screen for a place is shown below in figure X:

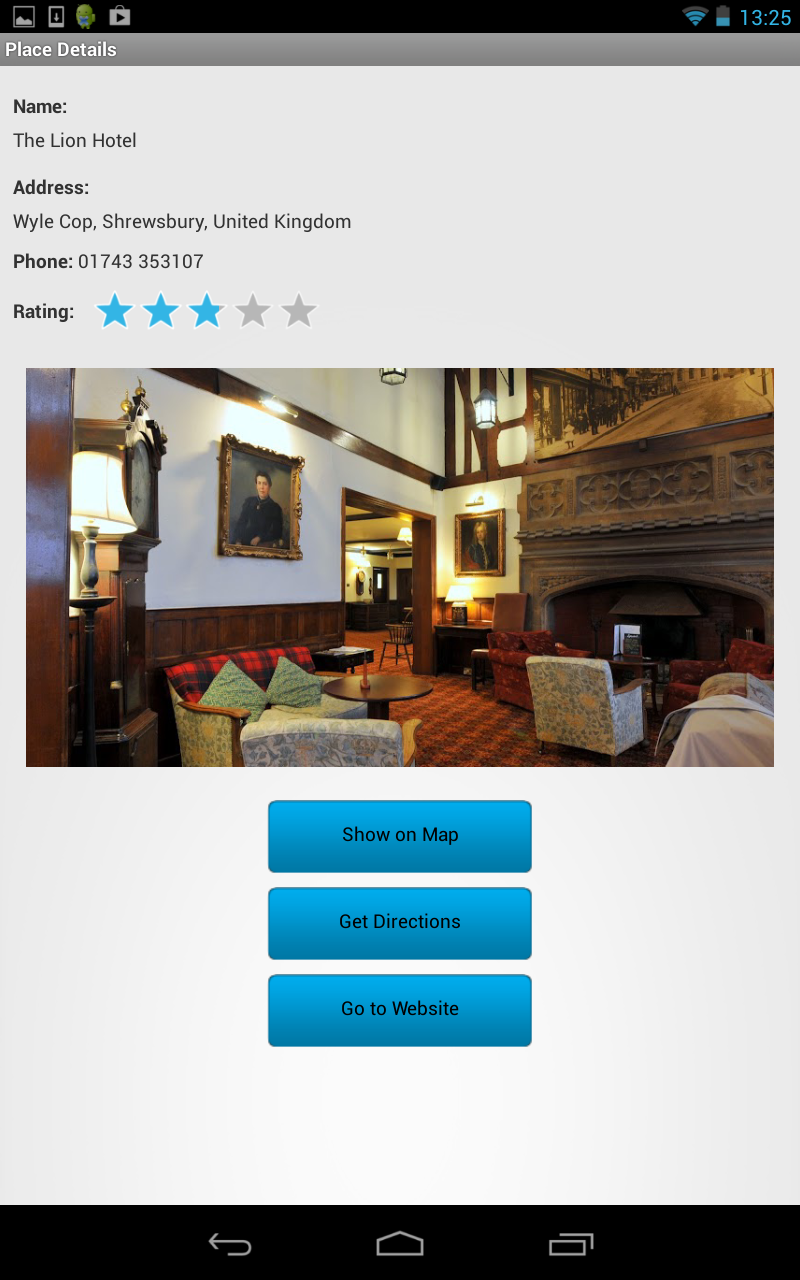


Fig. X. The place details screen.

# Implementation

As described in section two earlier in the report the development process for the implementation of the application developed into a hybrid process between several different types of development process. Originally it was anticipated that the waterfall model would be used, some elements of this were true as quite a lot of planning and design went into the project before the implementation started. However after it became clear that all parts of design and how the application would be tested could not be premeditated before implementation, the process became closer to a prototyping model to start with. Once the initial platform of the application was created it could be said that a feature driven development (FDD) model was then used in order to build and develop the additional features needed in the application.

In section 3.1 the overall architecture of the system is described. It states how that there are four major building blocks to the application, the GPS and Internet connection component, the graphical user interface elements, the Google Places search request modules and lastly the mapping elements. The decision was therefore made to create or find working prototypes of these four major elements and then to integrate them together and build on them to give a completed application that met the project requirements.

The first part of the application that was researched and targeted for development was the part that would make use of GPS to find the users current location. This element of the application would need to be able to function and find a devices location by using traditional GPS or a mobile phones network provider signal depending on what services were available on the device being used. In Android terms GPS is described as a fine-grained location provider where as a phone network is known as a coarse grained location provider. Firstly in order to start working on this part of the application section 8.1 of the Hello, Android book was read and gave a good basis for the work that needed to be completed. Following on from this it was though that it would be useful to find more about using GPS and location services on the Internet where a guide was found on the Android developers site [REF]. After reading this information a sample application for using GPS and Androids Location Manager was found in the Training section of the Android developer’s website. This was downloaded and run to see how it performed and was used. The source code was then analysed to see how it had been created and with what methods. From looking at the sample application, what needed to be done for this part of the application became apparent and focus could be turned to looking at the user interface design due to the confidence in being able to create the GPS module needed in the application.

The first piece of graphical user interface that was designed and developed was the MainActivity, both the controller and the view aspects i.e. the java file and then XML layout file. Due to the simplicity of the layout of this view this wasn’t too difficult to complete although Android can be quite awkward in allowing you to position things exactly as required. However developing the MainMenu activity was a quit a lot more difficult especially when the developer is new to developing for Android. Due to the design incorporating an icon based main menu the application needed to use a grid view. Although the layout file for the grid view is not enough on its own, a customer adapter needed to be written in order to display images within it. Again the grid view guide on the Android development website was really useful [REF]. The example that was given on the site really helped in understanding how this type of view worked and formed the basis of the activity that was created. Despite this, the basic activity created needed to be modified to include text labels for each icon. You would think this would be a rather easy task but it turned out this was not the case and it was more difficult to do than anticipated.

In order to display text underneath an associated icon, not only do you need to have a customer adapter but it was discovered that you need the adapter to inflate yet another layout file. The additional layout file (gridview\_row.xml) in the application is used to provide the layout for each individual tile in the grid (image and text). So effectively there is a layout file that is indirectly held inside another layout file to produce the final finished main menu view. The procedure to evaluate how this could be designed and implemented proved to be more time consuming than initially thought. Once the grid view had been successfully created the formatting of the grid also took some time to complete successfully. This is because the screen needs to display correctly on multiple screen sizes and devices. Therefore various column widths and vertical and horizontal spacing’s had to be trialled on the two devices I was testing on(section 5.2) to produce a layout that worked appropriately on both devices. Despite these time consuming issues the outcome produced was of a professional manner and created an aesthetically pleasing main menu independent of the device being used. This also meant that another objective for the project was met.

The next major segment that was researched and understood for development was the mapping aspect of the application. Before starting the implementation section 8.3 of the Hello, Android textbook had been read about embedding maps into an Android application to get an initial understanding of what needed to be done [REF]. To improve this understanding the documentation for Google Maps Android was discovered and read [REF]. However, unfortunately this aspect of the system became a setback in the project. It was discovered that version one of the Google Maps Android API had become officially deprecated on December the 3rd 2012 in the middle of the project .The API site also stated that ‘No new features will be added to Google Maps Android API v1’ and despite that apps using v1 will continue to work on device ‘Existing and new developers are encouraged to use Google Maps Android API v2’. This meant that most of the research and information gained about embedding maps in Android applications had become void. This was especially apparent when research on version two of the API was taking place, the way in which mapping works on Android has changed greatly although for the better as it allows much better and more interactive maps to be created.

Being able to create an Android application that uses the new Android maps API unfortunately involves quite a number of steps, and is a lengthy setup procedure [REF]. Firstly the Google Play Services software development kit had to be downloaded, installed and configured for usage on top of the Android SDK that had already been installed. The library project then had to be put into the source tree where the TourInf Android app project was held. This then allows the client library to be referenced for use in you Android project. Adding the library file to an Android is project is easily done in the Eclipse IDE [REF] where any libraries needed are simply added in the properties for the project.

Once these stages have been completed then a Google Maps API Key has to be obtained in order to use the Google Maps servers. This key is then used in your application so that when calls are made to the API that your credentials can then be approved. Again the procedure to get an API key was a lengthy procedure and more difficult to obtain than expected. It also delayed the making progress with the development and understanding of the mapping element of the application.

To get a map API key firstly you have to get your applications digital certificate known as its SHA-1 fingerprint. In order to do this you have to locate your keystore file and run a command in the command line to open the file and get the fingerprint which is the sequence of 20 two-digit hexadecimal numbers separated by colons that can be seen in the list. Once the SHA-1 fingerprint has been obtained you can then resister for the Maps API and others in the Google APIs Console found online [REF] although to sign in to the console you need to register for a Google account. Once you are logged in a new project has to be created for your application. You can then go to the services page and switch the Google Maps Android API v2 on by clicking a switch indicator. The last process is then to actually obtain the key. To do this you have to visit the API access page and create a new key using your SHA-1 fingerprint and your application package name. The API key is then returned and can then be used in your application where it needs to be declared in the AndroidManifest.xml file along with a number of permissions e.g. Internet, GPS, OpenGL etc.

After this process had been completed the sample app for the new API was imported into my Eclipse workspace from the Google Play Services SDK that it’s bundled to [REF]. The sample app demonstrates a number of different maps, what can be done with them and their features. After analysing the source code of the sample project it was decided to move on to looking at developing the part of the project that would use the Google Places API to gain the place information data needed by the application. This decision was made as it had been established how to create the maps needed and confidence had been accrued that this could be done at a later stage.

In the initial phase of research and development for the Google Places part of the application it was remembered that to use the API then this would also have to be switched on in the Google Developers console and a separate API key for Google Places obtained. Following on from this, research needed to be done on how the Google Places API was used. After reading the documentation for the API [REF] it became apparent how the web service provided worked and what web URL’s would need to be created in the application to request the data and be able to parse it back for usage and display in the app. However what wasn’t clear was the best method to make the connection and get at the data in an efficient manner that could easily be reused. To try and get some inspiration on how to go about the task, the web was searched for examples of using the Google Places API in an Android application.

After quite a lot of searching an example of an Android application that used the Google Places API was found on the android hive website [REF]. The website gave a detailed step by step tutorial of how the sample app using the Places API could be made. The tutorial was read with great interest and was extremely useful and helpful in understanding how the Google Places API could be used in the project. In academia it is taught to make use of works that are particularly useful and already available as long as they are correctly referenced and not licensed. Therefore in this case it was decided there would be no point in basically recreating another version of the code that was found on the android hive site. At this stage a decision was taken to use the tutorial to build the basis of the application as it successfully allowed you to create an app that used the Google Places API. In order to use the third party code provided by android hive, the code would have to be thoroughly understood to make successful use of it and to be able to make the many modifications necessary to integrate the other components needed with it.

The tutorial was followed with the source code being copied step by step into a new project that was set up in Eclipse and then run once completed. The Android application ran with the Google Places search returning places in the local area, although this was only for cafes and restaurants as the type to be searched had been hard coded into the program. The application also included a button that should have showed a map displaying the location of all the places that were listed. This feature was not originally thought of being included, but after seeing the idea it was thought that it should now be incorporated. Unfortunately this element of the tutorial code did not work as the application made use of the old version of the Android Maps API which uses map views. Therefore the API key that had been obtained and input into the project did not work as it could only be used with version two of the API. As stated at the start of the project, the project aimed to make use of the latest technologies. Also, earlier in the implementation (details above) the decision was made to use version two of the Android Maps API so consequently the mapping aspect of the tutorial application would have to be redesigned and recreated to use the new version of the API so the app could be future-proof in the short term.

Now all the major components had been covered by research and source code had been written or understood for these components. However all of these elements needed to be integrated together in order to create the application that had been designed. The first thing that needed to be done was to delete the classes and layout files that were not needed from the code produced from following the Google Places Tutorial e.g. map classes and extra layouts. Once this had been done the first thing to be integrated with the Google Places module was the user interface activities that had been partly written already so that the screens had a layout and the home screen forwarded the user on to the main menu.

However to get the expected results from the application the user interface needed to be further developed to integrate and communicate with the Google Place classes. The requirements for the application stated a number of the items that needed to be fulfilled such as:

* The user being able to use their current location
* The user being able to enter a location to search from
* The user should have a variety of categories to select from

Being able to use your current location to get nearby places had already been implemented however the option for the Places code to use different categories depending on what had been selected in the user interface was not catered for so this was then developed first. The application needed a type of place (from the allowed list by Google [REF]) to be associated with an icon image and its text. To do this a HashMap was used to hold a key value pair between the integer position of the image and the type value. Therefore when an icon was clicked on screen, its int position in the grid was used to get the associated place type to the icon. The place type then had to be sent in an intent to the MainPlacesActivity so that the type required could be used in the web service request rather than a hard coded place type being used. Once this code had been written, it was tested manually on the device to ensure it performed correctly.

The next item to be developed was for the user to be able to enter their location manually. For this to be able to work in the context of the app, the app would have to convert the location (via town/city name or postcode) entered into a geographic lat/long value. Fortunately earlier in the implementation when looking how to create the GPS module for the app an example of being able to do a similar thing but in reverse had been seen. The actual name of the process for doing this is called geocoding so the example that had already been seen made use of reverse geocoding. From this example it was found that thankfully Google has an API that can help perform geocoding operations.

It was decided that for design the geocoder should be implemented in the MainPlacesActivity as this is where the initial requests to the Places web service is made. In order to make use of the geocoder in that class any location that was entered on the home screen needed to be forward in an intent to the MainPlacesActivity via the MainMenu activity. The asynchronous task that was being used in the load places class with the MainPlacesActivity then mad to be modified so that it would use either the users current location or a location that had been entered depending on which option had been chosen by the user. If a location had been forwarded to the Activity in the intent to start the activity then the string location entered would be converted into a lat/long by the use of the geocoder. The values that were returned would then be used in the parameters to call the search method in the GooglePlaces class. Whereas if the current location option was chosen from the home screen then the activity would make a call to the GPSTracker service to get the devices lat/long location and use that in a search request instead. Again this was tested on a device to ensure the changes made to the program had been made successfully.

Now this had been completed the basis of a complete application was there, with the views created for each activity screen with a home screen, main menu, list of places and place details pages all working and being linked together. Therefore it was time to develop some of the more advanced features of the application that needed to be included. The place list and place details views both required maps to be displayed on pressing a button in the view and as a major aspect this aspect was tackled first. The view that displayed all the places listed on a map was developed first. Having an example of using the new API (sample provided with the SDK) helped greatly with using the new fragment based map activities.

The basic map was created with a loop being used to add a marker to the map at the location of each place held in the serializable class. When the map was tested however, on a list of places in Aberystwyth the result was not quite as desired. The map had decided to centre over just one of the markers meaning that not all the markers and locations could be seen on the map at once, which was the point of this particular view. In order to correct this, the map needed to be centre around the whole list of places and at an appropriate zoom level. To resolve this issue the average latitude and longitude values from the places held in the list had to be calculated and then what is known as the camera was zoomed in on this average lat/long value in order to centre the map around all the markers and making as many as possible visible to the user.

The fragment class that would show the location of a single place and launched from the place details screen could then be developed. The MapActivity class was implemented a bit more quickly than expected after already creating a similar class to display a map for all the places and the fact that the map layout file could be reused. The map to show a single place was also a bit simpler to create as it just needed to set up and centre a marker on the map at the lat/long value that was passed to it from the SinglePlaceActivity. Once this had been completed the application could then show both singular places and lists of places for a place type on a map. Following on from these map classes that had been developed and completed it was natural and made sense to continue with this theme and start working on the ‘get directions’ function as the next feature that should be included and implemented.

When trying to start implementing the directions feature to complement the maps already created another problem arose. It was found after more research that the new Google Maps API for Android did not include any built in methods to get directions between locations. In all honestly is must be admitted that this was slightly overlooked earlier in the project as it was almost presumed that a method would be included for this, especially in the new API. An example had also been seen previously in research for the project showing directions working in version one of the API. It transpired that there is a way to still programmatically get directions displayed on an Android Google map although it is very complicated to implement in a professional manner. The method found involved making a call to another Google API, Google Directions, passing the lat/long/values of the two points between which directions were needed. This would return yet another JSON query that would need to be parsed. For each direction entry (node) in the JSON returned, what is known as a polyline would need to be added as an overlay to another map. The text for each direction would also need to be received and displayed.

Several examples were evaluated and it was felt the outcome of other developers’ attempts to get this feature working was not particularly ideal. To get results to be presented in a well organised fashion would also have been extremely difficult and complicated. Another problem was it looks likely by implementing this solution Google’s terms of service for using map API’s maybe broken unless a paid for license agreement was made with Google. Section 10 [REF] of Google’s Maps terms of service concerns licensing restrictions with section 10.2 being particularly relevant stating the following:

‘10.2 Restrictions on the Types of Applications that You are Permitted to Build with the Maps API(s). Except as explicitly permitted in Section 8 (Licenses from Google to You) or the Maps APIs Documentation, you must not (nor may you permit anyone else to) do any of the following:

(c) No Navigation, Autonomous Vehicle Control, or Enterprise Applications. You must not use the Service or Content with any products, systems, or applications for or in connection with any of the following:

(i) real time navigation or route guidance, including but not limited to turn-by-turn route guidance that is synchronized to the position of a user's sensor-enabled device.’

For this reason further research was completed to try and find another way of being able to include the get directions feature in the application. Eventually a solution was found when looking at what could be done with done with Android intents [REF]. There are two methods in which directions can be obtained using intents. One way is to launch a VIEW Intent Action which launches a browser to a URL it is given, (Google Maps website) with the lat/long values being passed in the URL. The other option was to use a geo URI which opens the maps application to a given location or query. It was decided to use the first option for a number of reasons. Firstly the Geo URI scheme is not quite fully supported yet and is still under development. Therefore one benefit of using the browser intent method is that it should be supported on virtually all devices, even the older devices that may not have a built in map application. This means that the directions feature should always work. Another advantage was also discovered. When this type of intent is launched on a lot of devices, they are intelligent enough to launch the map app using the information instead of the browser if the map app is installed on the device. The result was of this solution was really pleasing as the result that became launched from the application gave a really professional appearance which was especially pleasing after the earlier setback.

In order to meet the initial requirements for the project all that needed to be done now was to include a button to launch a selected places website if the establishment selected had one. To do this a slight modification was made to the place class to hold a key for a website if one was found in the JSON that is parsed in the Google Places request. This then allowed a button to be created in the SinglePlaceActivity that would be set to visible if a website was present. If the button is clicked, again an intent is fired to launch the website in a web browser. Now this had been completed the functionality which had been originally set out for the application had been achieved. Fortunately however, despite this being achieved there was still some time left to continue enhancing the application before this final report needed to be started.

There were a number of extra features that were considered for improvements and a select few chosen to be implemented. The first extra feature to be added was the ability to call the place shown in the place details view if the establishment had a phone number. This feature was chosen as it’s likely to be useful to users as they may want to phone the place in question to get more information or make a reservation etc. Although there was a slight difficulty in implementing this feature, not all Android devices i.e. most tablet devices, have a sim card and can make phone calls. Therefore the feature should only be made available to devices with a phone enabled. How to detect this was researched on the internet and luckily a post on stack overflow was found that gave a solution [REF]. Making this check meant that the button for the feature would only be made visible if the telephone feature was found in the devices package manager. The new version of the application was run on both of the devices used for testing (section 5.2), with the button being displayed on the phone but not the tablet device, as desired.

It was detailed in section one that originally it was stated that if there were time at the end of the project it would be great to include a rating for places listed in the Google Place results. For that reason, that was the next extra feature to be included. Again for this feature to be added another key had to be added to the Place serializable class to hold the rating value for a listing if one were present in the JSON place information. Rather than just displaying a number on screen, it was though that it would look more professional and pleasant if the app displayed the rating using stars in a rating bar. In Android there is a standard component for this and the rating could quite easily be set, if no rating was given, then simply no stars would be show at all. The rating returned from the API are the average rating given to the establishments from users of Google +.

After completing the extra two features detailed above their was time to complete one more feature and the feeling was that it would be nice to include some more visual elements in the application to split up the textual data. Whilst looking at the JSON returned for detailed place requests it was noticed that some places have a photo reference available. This key could be used to go to a web address that contained a photo of the place selected. It was thought that users would really appreciate being able to visually see what some of the establishments they are finding information about look like as it would perhaps help them make a verdict on whether to make a visit to the place listing. For this reason it was it was decided that the ability to display a photo of places should be implemented, although the implementation of the feature was quite difficult.

Firstly another static class was added inside the place class to handle photos, and a key in the main part of the class was added for an ArrayList of photos to be held. Then the main coding needed to be done within the SinglePlaceActivity. To get a photo another request is required to be made to the places API web service as detailed in the API [REF]. The code to do this also needed to be written in another separate thread (another asynchronous task) as otherwise it would case the application to crash as it would stall the user interface while the request was being made. Fortunately asynchronous tasks had already been used in the project to make other call to different aspects of the Places API so background knowledge had already been learnt on how this would need to be implemented.

The use of another training guide on the Android developers site [REF] was again helpful in explaining how this could be done with the use of an example. Learning from this example the code was then developed to get a places image. However when the feature was tested it caused problems with the layout of the user interface as the images returned were all different sizes and further problems were caused when trying to correctly display the photos across multiple devices. To resolve these issues the ImageView in the layout file for the SinglePlacesActivity had to be modified. To ensure that the photos were always displayed at the same size for each place and screen resolution independent, a height for the view was set at 300dp(density independent pixels) and a ‘centerCrop’ scale type was set on the image as well as the usual layout gravity properties and margins. To get the images to display and fit correctly the layout had to be embedded in a scroll view so that when the content was viewed on a phone the content would scroll if need to ensure access to all of the features in the view.

Now this extra feature had been implemented and with not much time left for implementation it was decided that no more extra features would be included and concentration was turned to neatening up the appearance of the application to make it look as professional and aesthetically pleasing as possible. It is very difficult to display things neatly and accurately an Android any many addition layouts had to be embedded within other layouts to get rid of alignment issues and other formatting problems to get views to be presented properly on different screen sizes, resolutions and orientations.

Some things that were expected to be rather trivial were not so, for example trying to change the colour of a standard Android button. To do this you cannot just simply set a hexadecimal property to the button as it loses its properties and shadow. To implement it properly the background has to be set to a 9 patch image and a selector XML file had to be written to determine what colour the button should be when pressed and not pressed. An article [REF] that was found was used as a tutorial to implement these changes, with the blue and white images being used from the article in the app. Although the button style used in the tutorial was not created as using one would have been unnecessary. Now the application had been re-evaluated and its appearance tidied up and enhanced the Tourinf Android Application was now complete and ready for further testing.

# Testing

## Overall Approach to Testing

To test the TourInf Android application comprehensively and extensively a number of testing methods have been used in an attempt to make sure the application should perform correctly and as required. Firstly as you would expect for an object orientated project unit testing has been applied where appropriate using Android own framework. Despite the application being tested through the user interface throughout the project, these tests needed to be done more thoroughly to make sure the application was functioning appropriately from a user’s perspective. To do this, two different methods of testing were used. An expansive test table was created and tests were run manually on the device. Additionally, an automated test tool was used called Testdroid [REF] which uses a recording of user input put in the user interface to automatically navigate through the application testing all views, communication between activities and functionality. Finally some user testing was completed where random individuals were asked to have a go at using at using the TourInf application and fill in a questionnaire afterwards.

### Hardware

Being an Android Application, the application created should work on the large majority of devices that are capable or running apps. To test the app appropriately, it needed to be tested on the two major types of device; mobile phones/smart phones and also a tablet computing device. This needed to be done for several reasons. The first instance that comes into mind when testing on two different devices is how the graphical user interface of the application is going to be displayed and laid out. The application should display appropriately on both devices despite the large difference in screen size and screen resolution. Although more importantly, it must also be ensured that the functionality of the application is withheld on both devices. This was very important for the testing of the Tourinf application.

The application uses a connection to the internet in order to provide its features and many tablet devices are WIFI only whereas mobile phones and some tablets need to be able to operate using only their network provider’s phone signal as well as on a WIFI network. Therefore in order to test the application appropriately, testing was completed on a Google Nexus 7 WIFI only tablet device with a 7” high resolution screen and running the latest version of the Android operating system, and also a Samsung Galaxy Ace mobile phone with a 3.5” medium resolution screen running a much older version of Android (2.3.6).

### Android Unit Tests

Unit tests were written for the application where appropriate to ensure that individual units of source code written for activities were fit for use. The developer of the project had written unit tests before in the form of JUnit. However this framework could not be used as despite the source code being written in Java it obviously contained Android specific methods and libraries meaning that it could not be tested using normal JUnit. This meant that other forms of unit testing would need to be discovered that could work with Android source code. Thankfully the Android developers’ website offers a lot of advice on how this can be done with a testing fundamentals page being especially useful in initial understanding [REF].

The Android testing API is still based on the JUnit API but it has been extended with Android-specific testing classes. However unlike normal JUnit test cases there are different categories of tests that have to be used in order to test different types of components. For example it has component specific test case classes for activity testing, content provider testing and service testing.

It was discovered that the approach to testing was also slightly different to normal JUnit testing. In Android testing, a test project has to be set up which links to the project being test and has to contain a test package which holds individual test cases. The Android SDK uses test tools to load the test package and the application project under test, and then executes an Android-specific test runner [REF].

For this particular project the main concern was testing that activities implemented were fit for use from a system perspective. However not all classes and activities were needed to be or could be tested. There are a few instances of this such as the map views/ fragments. There is no component specific test case in Android which could have tested these classes. There was also no point in testing the ConnectionDetector or GooglePlaces classes as their methods are called from other classes that were to be tested. Again the ImageAdapter class didn’t need testing as it is an adapter and essentially just sets information. While the serializable classes (Place, PlaceDetails and PlacesList) were not tested individually as they just hold data and therefore need to be tested through another activity as well.

In the end this meant that six unit test cases were created in the test package (Tourinf.test). Five of these tests were for activities whilst another tested a service (GPSTracker). The list of activities/services tested is as follows; AlertDialogManager, GPSTracker, MainActivity, MainMenu, MainPlacesActivity and SinglePlacesActivity. The last two tests in the above list are the most important as those activities test the large majority of features and functionality. The results of these tests can be seen in appendix X where it is shown that all the tests that were written pass successfully.

### Manual User Interface Testing

Although the TourInf application created was tested throughout the development cycle through the use of the user interface, to ensure the application was completely fit for purpose an extensive test table was created. The test table created tests every aspect of the application from a user perspective and details the actual and expected results of the tests that were carried out. The full testing table with results can be found in appendix x.

### Automatic User Interface Testing

To test the functionality of the user interface and the communication between activities it was felt that another method of user interface testing should be carried out. In order to do this effectively and in a manner that was different and complimentary to the user interface testing that had already been completed, these tests would be done automatically.

There were a number of different automated testing tools that were evaluated for the testing of the project. These tools were the MonkeyTalk tool from Gorilla Logic [REF], Testdroid [REF], its recorder and cloud service, and lastly Androids own monkeyrunner tool [REF]. All the automatic testing tools that were evaluated appeared to be useful tools but after reading and analysing what all the tools could offer, the Testdroid tool was selected to be used for the project.

The monkey runner tool provides an API for writing programs that control an Android device from outside of Android code. Monkeyrunner lets you write a Python program that installs an Android application or test package, runs it, sends keystrokes to it, takes screenshots of its user interface, and stores screenshots on the workstation used [REF]. The monkeyrunner tool was not used for the project as it seemed much more difficult to use than the other tools on offer, having to write Python programs to run the tests. To do this the Python language would have to have been learned. In addition to this the only evidence of a test passing successfully is screenshots which can be taken. Screenshots would not necessarily prove that the application is performing acceptably and all functionality working as desired.

The MonkeyTalk automatic testing tool uses a graphical user interface to allow the user to write tests almost in a spreadsheet test table manner using their own simple scripting language statements or JavaScript. The program looked quite easy to use however the more advanced features didn’t seem very well documented on their website or demonstration video. The website also didn’t display in detail what conformation you get returned about a test that has passed. These details being missing helped make the decision to use Testdroid.

Testdroid was the automated tool selected for usage. The software on offer was far easier to use than both the competitors described above. Test cases were created simply by recording user input and keystrokes in the application itself. No programming or scripting was required as the software automatically reverse engineers the recordings created to then create JUnit test code by using the Robotium framework. Robotium is an open source test framework for creating automatic test cases for Android applications. It uses code similar to Androids own for unit testing but with extra functions for user interface testing. These benefits were extremely useful in testing as because the tool was easy to use, it was easily understood and tests were created with few problems. Furthermore the program produced source code tests of the official standard, JUnit, which is the test framework that had been used earlier in the project for unit testing albeit in an Android specific version. The results of the tests created in Testdroid using the Testdroid recorder can be seen in Appendix X.

Testdroid also lets you try using one of their complementary products to the Testdroid recorder, Testdroid Cloud when taking out a free trial. Testdroid cloud [REF] is another really helpful tool/service in automated testing. It allows you to upload your test package online to their website where you can then select a range of devices for which the test cases uploaded will be run on. This allows you to test the application across multiple types of devices and API levels quickly and easily, and most importantly without needed to have or own the devices. The Testdroid cloud also lets you use its ‘App Crawler’ service. This type of test exercises the application systematically by advancing the application through each of its views held in the view hierarchy. The test results from the Testdroid Cloud and its App Crawler test can also be seen in Appendix X.

### User Testing

To ensure that the application developed was fit for use and met potential user’s needs and expectations, some user testing was completed. To do this a questionnaire was created which asked the users simple questions about what they thought of the application, what comments they had about it, and how it could be improved. The questionnaire was given to many different types of user e.g. different ages, computer competency, genders etc to try and get an unbiased view point. The questionnaire created and its results can be viewed in Appendix X.

# Evaluation

* Were the requirements correctly identified?

Now the project has been completed we can now assess whether the requirements of the application were correctly identified at the beginning of the project. Looking at the original list of requirements it appears that in the large majority of cases that they were indeed correctly identified. This is due to the fact that the requirements for the project were stayed stable throughout the project as the functionality laid out for the application was accurate meaning that requirements didn’t need to be changed and were as expected.

However on the other hand there was a requirement that was incorrectly identified. The requirement that was set out stated that ‘each category of tourist information will have a sub category if applicable e.g. for food; Chinese, Indian, Thai, Japanese’. Unfortunately after doing some more in depth research on the Google Places API after the requirements for the project were set, it became apparent that implementing this sub category feature would not be possible using the clean design of the system. The Google Places API has a list of supported types for place searches and this list does not include any of the sub category’s that we had hoped would be used in the application, such as those listed in the food example above. There is a workaround for this to create a solution however it involves keeping a search category for each subcategory. Unfortunately there was not enough time for this to be researched in any detail or implemented.

* Were the design decisions correct?

System Design

User Interface design

Reflecting back on the user interface design it appears that most of the design decisions were correct. In order to achieve the aims and objectives of the project, the application needed to be easy and simple to use for the user. Ultimately the design that was created has ensured that this objective was met and making the app fit for purpose. The decision to use an icon based menu was particularly appreciated by users of the system in testing as they stated that it was a lot quicker to use because of this feature and also more aesthetically pleasing. This confirms the feeling that good decisions were made in the design of the user interface.

However despite this, the user interface design could still be improved from the good basis that’s already been built. Without the time constraints to complete the project it would have been nice to have gone ahead with the original list item designs developed earlier in the project (figure …) where the list has expandable icons to show place details. Unfortunately being able to display the information in this way is very complicated. The ability to show an icon image of the place to the side of the list which could be clicked on and enlarged would be another great feature that could be added to the existing interface.

* Could a more suitable set of tools have been chosen?

In general the tools that were selected to complete the project were well suited to what the project was aiming to achieve and were fit for purpose. The tools used were extremely useful in the development of the application. The Eclipse IDE with the ADT plugin installed was especially helpful and useful throughout the development process as it has been designed to be able to handle the development of Android applications. The user interface part of the development environment was extremely helpful as it allowed you to preview what the XML layout files would look like when the application was launched without actually having to launch it. This meant that changes could be made and the results of these changes seen much quicker, allowing for more efficient development. In this instance I don’t feel that any other development environment could have offered as much to the project as the Eclipse package.

The choice to use GitHub to backup the project was also a successful one. Again the support tool saved time and effort by making the development process faster. The GitHub for Windows application is a great extra add-on to help keep your repository in sync. I could have used another subversion product however it is unlikely that any other service would have offered anything with any additional benefits, whilst the bonus of having the Windows application suggests it was the best choice in the context although it is understandable that some developers are put off by their repository’s been displayed publicly.

* How well did the software meet the needs of those who were expecting to use it?

After analysing the responses of users in the acceptance testing of the application it became clear that in most cases the software met the needs of those that would be interested in using such an application. In general the users were impressed with how quickly they could operate the application and find the essential information about local and distant places. They liked the general layout and organization of the application and how the process of finding different places was logical and allowed them to easily drill down through different screens to get more detail on the items they had selected. The large majority of users also expressed that they thought that the mapping features of the application were highly impressive and were very useful in operation due to them being highly interactive allowing them to use gestures to manipulate the map how they wanted to.

Then again the users of the application did comment that the application could look more aesthetically pleasing to the eye. Whilst they liked the simplistic nature of the app and the clean minimalistic design, they also felt that more colour could have been used throughout the application. Another feeling was perhaps the application had too much white space available that could have been taken up with additional content or a layout that made better use of the additional space. The main observation made though was regarding the appearance of the icons. The view was that the users liked their simplicity and the fact that they could easily identify different categories through the use of the icon images. However again the opinion was that perhaps the icons could have been different colours, contained colour or had different colour backgrounds.

* How well were any other project aims achieved?

For this project not only did I wish to create an application but I also wanted to learn about developing for Android and mobile devices in general. Gladly after completing this project I feel that I have learnt a great deal about developing for Android and how it differs from a usual software project. I feel that I have a good understanding of how these types of applications are built and how my own project can be improved. I have enjoyed completing the project and plan to continue developing the application that has been created in my own time to include the extra features I would have liked to have added and continuing to improve the skills that I have gained throughout the project.

* If you were starting again, what would you do differently?

If I were to start the project again I would do a number of things differently. One of things that I would change is my approach to the project. At the start of the project I spent quite a lot of time researching about Android apps and how they are structures and developed. The knowledge that I gained was rather useful throughout the project, however looking back to that time, I was perhaps looking at the area in too broad a perspective. In the end a few of the topics and things that were learnt initially were not needed for the project or not needed until much further down the development cycle. This meant that unfortunately my time was not used as efficiently as it could have been to create the app on the project. It meant that some of the things I researched and read about at the beginning of the project I had to revisit and go over when I was developing those particular parts of the application to remind myself about certain aspects of design and implementation.

If were to start again I would have spent less time doing my initial research and I would have started developing parts of the application earlier. In most cases things would only be researched when the need to use a particular item or tool was indentified and when any errors occurred within the application that could not be resolved reasonably quickly. Ultimately the time spent doing research could have been utilised better and may have enabled me to add some of the extra features that I would have liked to have included in the completed application.

I am very pleased with the outcome of the application that has been developed and what I have achieved. However after completing the application I have a thought of quite a few ideas for how the app that has been created could be improved in the future with a number of features that I would have liked to have included in the application but sadly did not have time to complete. One such feature would be for the app to display the opening hours/times of any shops and business’ which had them readily available on their Google Places listing. I would have also have liked to display a simple indicator on the user interface to show whether the establishment is open at the time the user looks at the item. I think this would have been another good feature and would have enhanced the functionality of the app. Users would have found this useful as they do not want to make a trip to visit any of the listings that are not open at an appropriate time.

Another more advanced feature that I would have like to have developed is the inclusion of reviews in the app. Some listings have been reviewed on Google Plus and this information is returned in the data that is returned by the Google Places web service. It would have been nice to of been able to pull out and display this information in the app to the user to give them an indication on what the respective place is like and let them get a clearer indication on whether they would like to visit the establishment. In addition to this it would have been really good for users of the app to have been able to add reviews of the places listed themselves so other users of the app could read their feedback. This would have been quite a complicated feature to implement as it would require the use of a database for storage that would also need to be hosted.

The ability to upload photos is another feature that I would have liked to have added to the application. Recently users of applications are not just using them to access information, they also enjoy returning information back to the application or to somewhere else on the Internet. Apps and social network sites such as Instagram and Twitter have made uploading images of places popular in our society and day to day lives. This makes me think that the ability to upload photos on the app would be appealing to a large number of users. It would allow users to take photos of the places and business’ that the app lists and also of any of their products or services if applicable, for example someone could take a photo of the exterior of a restaurant and also the food that is available. Such items would be useful to other users of the app, again helping them make a better judgment on making a visit to an establishment or organisation.

This brings me on to the link with social media. It would be fantastic if the application could be linked into users’ social networking profiles to allow the application to fully integrate social network features as these sites are increasingly being used by the general public and commercially by companies. When a user is placed within the vicinity of a place or uploads a photo or review of a listing this information could be displayed on social network sites. Not only would this be a good feature for application users to make use of, but it would also help promote the app and its brand.

Lastly I have also thought about how I could make the application commercially viable and how the app could be monetised. One obvious way of generating money from the application is uploading the application to the Google Play Store and charging users to download and use the application. This is unlikely to be a popular decision with most users as currently the majority of apps on the Play Store (especially utility apps) are free to download and use. It is worth noting that to release the application on to the play store a lot more work would need to be completed, notably more testing e.g. different Android API versions and a greater range of different sized displays. The usage agreements of the Google Places web service and Google Maps API would also need researching in much more detail. Google do have usage limits for obtaining the data that has been used and these limits are likely to be exceeded if the app was being used by many individuals after being made available to download on the play store. Google are also likely to charge for this data due to the large amount of usage. Despite this, once these things had been fully analysed and evaluated it would be great to get the Tourinf app on the Google Play Store for Android users to download and make use of.

Additionally, what I also would have liked to have implemented to bring in revenue for the application is the ability to show advertisements in the application. These adverts could have perhaps been shown before the app has launched and also displayed along the bottom of the screen or elsewhere. Google have already created a tool available on the market that allows mobile app developers to easily add advertisements into their application. This tool is called the Google AdMob Ads SDK. This kit allows the app to show readymade advertisements to the user with very little code needed to be written by the developer and embedded in the app. The owner of the application will then receive revenue when the adverts are clicked on and visited. I feel that using this tool/kit from Google would have been a great way to quickly and efficiently monetise the application in a professional manner. It is also a method that would be unlikely to cause users to stop using the app or not download it in the first place, unlike making them pay for the application initially which often puts potential users off. In most cases I feel that users aren’t too concerned about having advertisements inside their apps as long as the app is free to download and they aren’t too annoying and take up a large proportion of the screen. I therefore believe that using the AdMob Ads SDK would be a very good way to monetise the application developed and would have liked to have integrated it into the application had the development phase been longer.

* Improvements
  + Opening hours
  + Photo’s- got from places API but not from User
  + Reviews- Textual
  + Call- Know whether the device has a phone enabled
  + Social network integration
  + Commercial-App store, Adverts-Google Adsense or Google AdMob Ads SDK
  + Remember to say that I’m pleased with the outcome and happy that I managed to add some extra features to what was originally planned.

# Appendices

* 1. Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

* 1. Original Graphical User Interface Design

1. 5.

**Use Current Location**

Enter City, Town, Post Code...

**TourInf**

**TourInf**

Honoured Guest

Address Lines

Address Lines

01970 123 456

**-**

Hot Dumplings

Address Lines 01970 123 456

Kam Sing

Address Lines 01970 123 456

**+**

**+**

Website

Map

Directions

2. 3. 4. **4.**

**TourInf**

**+**

Honoured Guest

Address Lines 01970 123 456

Hot Dumplings

Address Lines 01970 123 456

Kam Sing

Address Lines 01970 123 456

Mandarin

Address Lines 01970 123 456

**+**

**+**

**+**

Seafront Palace

Address Lines

**TourInf**

Food

Shops

POI

ATM’s

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Can’t find it...Search here

**TourInf**

Chinese

Indian

Thai

British

**TourInf**

**+**

Honoured Guest

Address Lines 01970 123 456

Hot Dumplings

Address Lines 01970 123 456

Kam Sing

Address Lines 01970 123 456

Mandarin

Address Lines 01970 123 456

**+**

**+**

**+**

Seafront Palace

Address Lines

**TourInf**

Food

Shops

POI

ATM’s

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Example

Can’t find it...Search here

**TourInf**

Chinese

Indian

Thai

British

* 1. Class Diagram
  2. Testing

**Unit Testing**

**Testing Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.** | **Test Details** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | Home screen displayed correctly. | Displayed correctly | Displayed correctly | Pass |
| 2 | Use current location button | Forwarded on to main menu, category selected lists places relative to user’s location. | Main menu displayed. Correct places for user’s location found. | Pass |
| 3 | Enter location | Forwarded on to main menu, category selected lists places relative to location entered. | Main menu displayed. Correct places for location entered found. | Pass |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |

**Automated Testing**

**Testdroid Recorder**

**Testdroid Cloud**

**User Testing: Questionnaire**

# Annotated Bibliography

1. Google Places API. Various: <https://developers.google.com/places/>
2. Google Maps API. Various: <https://developers.google.com/maps/>
3. Some recent evidence to support my theory that tourist information centre visitor numbers are in decline. <http://www.frasercoastchronicle.com.au/news/visitors-at-centres-in-decline/1615621/> . Further evidence that tourist information centres visitor numbers have been declining for some time, a council report. <http://www.eastriding.gov.uk/public_reports/TheCabinet/16March2010/Tourist%20Information%20Centres%20-%20Hornsea%20and%20Withernsea.pdf>
4. An article which discusses how the Google Play Store is rapidly catching up with number of apps available on the apple app Store and how it could overtake it in the near future. <http://news.cnet.com/8301-1035_3-57521252-94/can-apples-app-store-maintain-its-lead-over-google-play/>
5. Manchester Tourist Guide Android Application on the Google Play Store. <https://play.google.com/store/apps/details?id=com.mymobilemanchester&feature=search_result>
6. Pocket Britain Android Application on the Google Play Store. <https://play.google.com/store/apps/details?id=com.phonegap.pocketbritain&feature=search_result>
7. Yell Android Application on the Google Play Store. <https://play.google.com/store/apps/details?id=com.yell.launcher2&feature=top-free>
8. Ed Burnette, Hello, Android: Introducing Google’s Mobile Development Platform, Third Edition, for Android 2. Pragmatic bookshelf 2011. A book lent from a friend to research programming for Mobile Devices.
9. The strategy and theory of using Android’s Network Location Provider together with GPS to determine a user’s location. <http://developer.android.com/guide/topics/location/strategies.html>
10. An overview of what Grid View is on the Android Platform and how you can use it. <http://developer.android.com/guide/topics/ui/layout/gridview.html>
11. The Android plugin for Net Beans. <http://kenai.com/projects/nbandroid>
12. IntelliJ Idea IDE with support for Android. <http://www.jetbrains.com/idea/features/android.html>
13. The download page that enables you to get the Eclipse IDE, Android SDK and ADT plugin. Although I didn’t use the bundle as I had Eclipse already installed. <http://developer.android.com/sdk/index.html>
14. ‘Building Your First App’ Android tutorial. <http://developer.android.com/training/basics/firstapp/index.html>
15. Location where some of the University Android worksheets and presentations can be found. <http://users.aber.ac.uk/cwl/workshop/>
16. Mobile Device operating system market share figures for the 3rd quarter of 2012, showing iOS’ and particularly Androids dominance. <http://www.gsmarena.com/android_and_samsung_build_on_their_market_shares_in_q3_2012-news-5082.php>
17. Shows that you can only develop for iOS on an Intel based Mac. [http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios\_development\_workflow/45-iOS\_Development\_FAQ/faq.html#//apple\_ref/doc/uid/TP40007959-CH12-SW1](http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios_development_workflow/45-iOS_Development_FAQ/faq.html)
18. Reveals methods of using Mac OS X on a non-Apple PC. <http://en.wikipedia.org/wiki/OSx86>
19. An article about the negative feedback and complaints made about Apple’s new Map app. <http://www.bbc.co.uk/news/technology-19659736>
20. News article stating the possibility of a new Google Map app being launched for iOS 6. <http://www.pcmag.com/article2/0,2817,2412213,00.asp>