DUBLIN INSTITUTE of TECHNOLOGY
Institiúid Teicneolaíochta Bhaile Átha Cliath

Activity Recommendation For Android
Final Year Project Report

DT228
BSc in Computer Science

Lee Murray
Edina Hatunic-Webster

School of Computing
Dublin Institute of Technology

March 2015

Abstract

Sometimes a person can find himself or herself at a loose end with nothing to do. This could be due to a number of factors including: The weather, the time, or a lack of information on the places and activities that surround them.

This project is a recommendation system that can recommend a wide range of activities to a wide range of people. A number of things are taken into consideration before any recommendations are made, such as the current weather, the person's location and a number of other specified parameters that are set by the person

This project aims to achieve an application that can recommend an activity for anyone, anytime and anywhere.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Lee Murray

25th March

Acknowledgements

First of all I would like to thank my parent's Judy and Doug for their continued support over the last number of years. They encouraged me to stick with Computer Science at times when I felt like quitting. Looking back now, I have only them to thank for where I am today. Secondly, I would like to thank both of my grandparents, especially my grandmother Nina, for her uplifting words of wisdom and encouragement she has conveyed to me for as long as I can remember. Thirdly, I would like to thank my supervisor Edina-Hatunic Webster for her advice throughout this project. Finally I would like to thank my extended family and all of my friends for their support, especially my Aunty Antoine for proof reading my dissertation.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF CODE SNIPPETS

Chapter 1 - Introduction

This Android application will recommend 'activities' pertinent to the user based on certain criteria. The application automatically collates the user's location, weather conditions and current time and the user specifies their gender and age. Then, by adjusting some parameters within the app the user indicates how; Active, Adventurous, Sporty, Daring or Cultural they would like the activity to be. The activities are then returned to the user and displayed on a map.

1.1 Project Overview

The user will download the application from the Google play store to their android mobile device or tablet. Once the user opens the application they will be faced with the 'Home Screen' where there will be two options: 'Create Account' or 'Sign in'. If the user presses 'Create Account' they will be brought to a new screen. There they will choose a username and enter their name, email address, country and a password that must be confirmed. The information is then sent to and stored in a database- the password is hashed before it is inserted in order to provide some security.

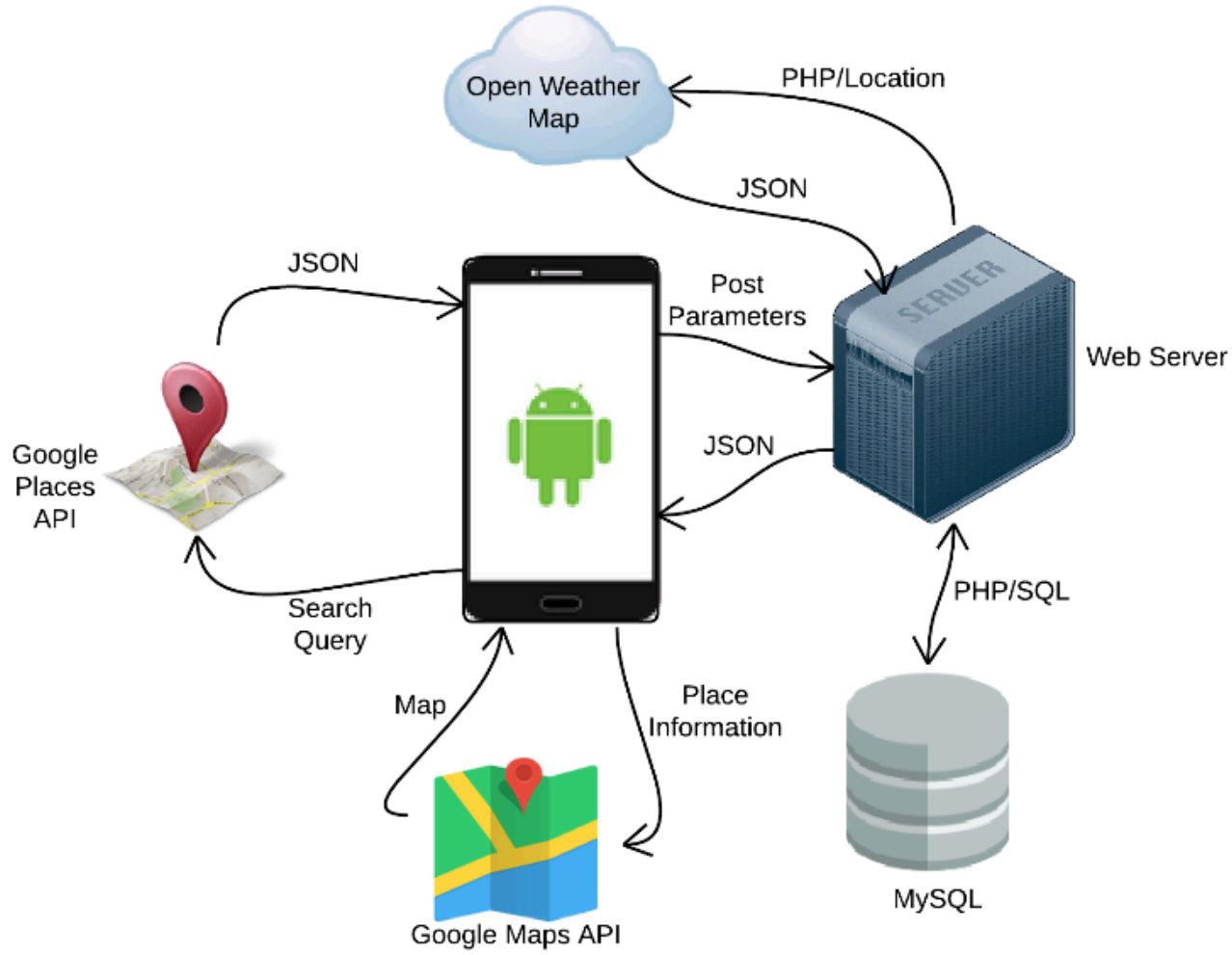
When the user signs in to the application they will be brought directly to the 'Home Screen'. There they will see the current time displayed (at the top of the page) along with some weather information underneath. If they tap on the weather information they will get more details such as pressure, wind speed and direction, on a new page. Under the weather on the 'Home Screen' the user will see a large colourful button, which reads: Get Activities and below this two more buttons that read: Sign Out and Settings respectively. When the user clicks on the 'Settings' button they will be brought to the Settings page, where, their custom settings will be loaded from their previous session. If it is the users first time to the Settings page then some default settings will be implemented. On the Settings page, the user will be able to specify their gender and age, the search distance, and how active, cultural, adventurous, sporty or daring they want the activity to be. When the user presses 'Submit' on this page their settings are saved to the database and they are brought back to the 'Home Screen'.

Once the 'Get Activities' button is pressed, both the current weather and the settings specified by the user, is taken into consideration. A Google map will be displayed using markers to show the location of the recommended activities. The user can press any of these markers to get some more information about the activity.

When the user presses the 'Sign Out' button they are brought back to the Home Screen where they can sign back in or create another account.

A lot of the coding for this app was originally done in Eclipse Juno. However with the stable release of Android studio, the project was migrated to that IDE for completion.

Figure 1: Technical Architecture Overview



The technical architecture for this project can be seen above. This Application uses a Web Server, a MySQL database for storage, Google Maps API to display a map, Google Places API to get activities and Open Weather Map to get the weather. This will be expanded on in the Design and Architecture Chapter.

1.2 Project Background

The idea for creating a mobile app based on activity recommendation was enhanced by an idea a lecturer had concerning a weather based e- marketing platform for business. While working on other features and extra functionality, the idea of weather being such a determining factor in people's decision -making struck a chord. That the weather should be taken in to consideration when recommending an activity to the user became a feature. However, the weather was not going to be enough on its own so user's age, gender and location were added to the criteria. Next, an idea arose to ask questions to the user in order to help the system learn a little more about them. This later evolved into the user being able to use slider bars to adjust parameters including: how active vs. lazy the activity recommended should be or how cultural the activity should be.
Finally, 'Time' was implemented into the decision making process so that the system

would not recommend an activity to the user if it wasn't possible at the time e.g. the cinema at 3am.

1.3 Project Aim and Objectives

The aim of this project is to design and develop a fully functional mobile application that can be used easily by users of all ages.

1.3.1 The Most Important Criteria

- User interface Design
- Design of the system
- Development of key features
- Completeness
- Usability
- Testing and feedback

1.3.2 Deliverables

- Dissertation
- A fully functional mobile application

1.3.3 Priority Features

- Account Creation
- Signing In
- Weather retrieval
- Activity Recommendation

1.4 Document Structure

1.4.1 Chapter 2 – Background Research

In this chapter all of the research done for this project will be showcased and discussed. The mobile operating system market share will be analysed and documented. Several applications with similar functionalities to that of the proposed project will be examined. Mobile platforms and integrated development environments will be evaluated and one will be chosen for the project. A number of Programming languages will be looked at and one will be chosen for the core development of the proposed application. Database technologies will be evaluated and one will be chosen for the proposed project. Necessary API's and expert systems and scripting languages will also be documented.

1.4.2 Chapter 3 – Architecture and design

The design and architecture chapter will begin by discussing the chosen software methodology that was used in the development of this project. The functional requirements will be discussed and illustrated by diagrams with aid from the unified

modelling language (UML). The database design will be illustrated and described. The technical architecture for the project will be displayed.

1.4.3 Chapter 4 – Implementation

The implementation chapter will outline how the project was implemented from the beginning until the end. From the layouts and user interface, to the creation of accounts and recommendation of activities.

1.4.4 Chapter 5 – Testing

Both black box testing and white box testing will be carried out in this chapter. For black box testing the application was given to 4 users to test along with a questionnaire in which they filled out. For white box testing a number of test cases were created.

1.4.5 Chapter 6 – Project Evaluation

The original project plan is reviewed. Challenges faced during the development of the project and how they were overcome are discussed along with the learning outcomes.

1.4.6 Chapter 7 – Conclusion

A short summary of the document and some future work is discussed.

Chapter 2: Background Research

2.1 Similar applications

During the initial research for this project there were searches done to find similar applications to the proposed app. The following three systems were of relevance to the proposed project and had certain commonalities.

2.1.1 Weotta Go

Weotta Go is an iPhone app that recommends activities based on the users location and the time of day. "Its recommendations are delivered as a stack of photos, which you can tap on for more information, drag down to save in a list, or swipe across to say that you're not interested. As you do that, the list will change to show you more items in the categories that you're interested in and less of everything else.[1]"

Figure 2: Weotta Go Screenshot[1]

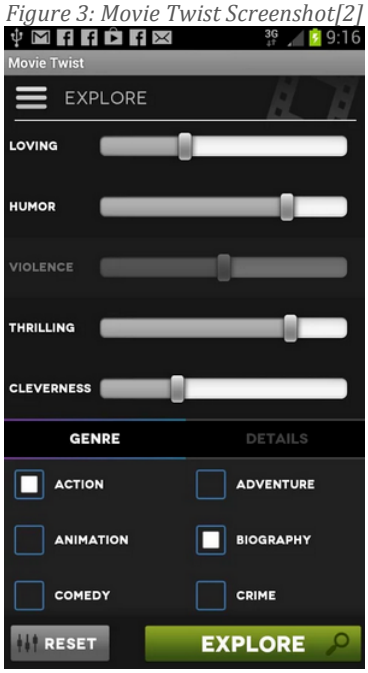
Weotta Go isn't available to download within Ireland so it was not reviewed hands on. It would seem as if Weotta Go takes the time and users location into consideration and also learns through a like/dislike style system. The time being taken into consideration is one similar feature that this application proposes implement.

2.1.2 Movie Twist

Movie Twist is an android application that recommends movies to the user. The user is able to enter in a favourite movie of theirs in order to get recommendations of similar movies. They also are able to use the 'explore' tool in order to adjust different

parameters using slider bars. These parameters include: loving, violent and thrilling. The user can also search the app based on their mood. The app will always remember the user's favourites from the past so it can learn the best movie to recommend next time.[2]

Figure 3: Movie Twist Screenshot[2]



In figure 3 above the 'explore' page is on display. There are a number of seek bars that allow the user to adjust different parameters that affect the outcome of the movie to be recommended. This feature is similar to what is to be included in the proposed application.

2.1.3 Skymosity

Skymosity is a weather-marketing platform. It allows a business to gain actionable insight on how every weather condition and temperature range influences purchase behaviour and product trends on their website. "With Site Targeting from Skymosity, you can dynamically serve different content based on each visitor's weather condition. Seamlessly create dynamic site content, site overlays, and more when a visitor meets certain campaign conditions.[3]"

Skymosity is a great form of e-Marketing that takes full advantage of the weather, to target the consumer with the most appealing products, based on their research.

2.1.4 Similar Systems Conclusion

The three systems just discussed each have their own similarities to the proposed project. Weotta Go takes the time into consideration before making a recommendation. Movie twist allows the user to adjust different parameters that in turn harness different recommendation outcomes. Skymosity uses the weather to make certain e-marketing decisions and the proposed application will also take the weather into consideration when making a recommendation for an activity.

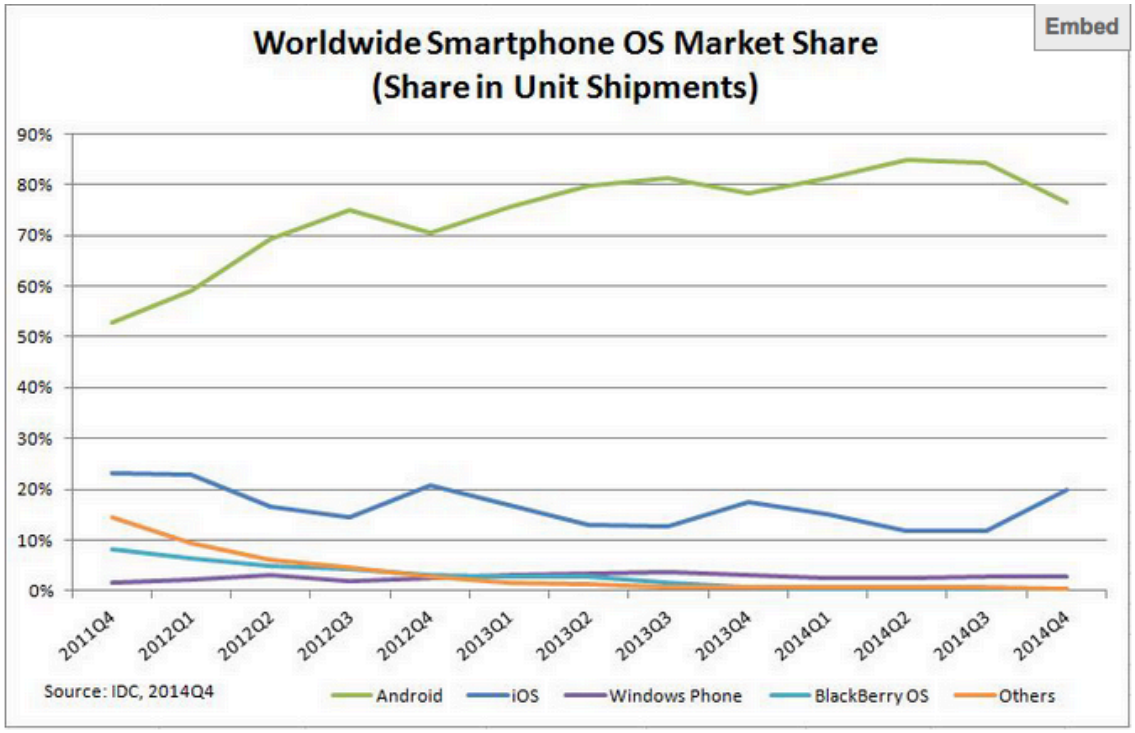
2.2 Mobile Operating Systems

2.2.1 Mobile OS Market Share

It is important to know the market share for mobile operating systems (OS) when developing an application for the mobile market. Android is by far the leader of the smartphone OS market and has been since its release. Apple's iOS is in 2nd place which is impressive as it is restricted to just the Apple iPhone. Android is the OS for many smartphone manufacturers such as Samsung, HTC and Google. Figure 4 illustrates the smartphone OS market share from the 4th quarter of 2011 to the 4th quarter of 2014.

By looking at figure 4, it is evident that android is by far the most popular OS globally. Owning about 76% of the smartphone OS market towards the end of 2014. iOS is in 2nd place owning about 20% of the smartphone OS market by the end of 2014. This information informs us that together Android and iOS dominate the smartphone OS market share owning 96% of it combined, while Windows Phone, Blackberry OS and others own the remaining 4% of the market.

Figure 4: Worldwide Smartphone OS Market Share 2011 – 2014[4]



To conclude, the best mobile OS to develop an application for is either Android or iOS, there would be no point from a entrepreneurial point of view developing a stand alone application for Windows Phone, BlackBerry OS or any other smartphone OS.

2.3 Technology Research

2.3.1 Mobile Operating Systems

2.3.1.1 iOS

"*Before you start writing code, you'll need to do some housekeeping. First, you'll need to install Xcode, Apple's development environment, as well as the iOS SDK. Both of these are available directly from Apple via the Mac App Store.*

However, if you want to distribute your applications, or even just deploy them onto your own device, you will need to register with Apple as a developer and then enrol in one of the developer programs. You'll then need to create, download, and install a number of certificates and profiles to allow you to deploy your applications onto your iPhone, iPod touch or iPad. [5, p. 7]"

When developing in Xcode the language that has been primarily used since 2008 is objective C. Recently however, Apple has released a new language called Swift. It has a concise yet expressive syntax and the apps run lightning fast. Swift can also run alongside objective C, allowing current developers to integrate it into apps that have already been developed. New developers can utilise the two languages right from the very beginning.[6]

2.3.1.2 Android

The Android mobile OS is used by many smartphone manufacturers including Acer, Asus, HTC, Huawei, LG, Kyocera, Motorola, Samsung, Sony and ZTE.[7]

"*One benefit of developing Android apps is the openness of the platform. The operating system is open source and free. This allows you to view Android's source code and see how its features are implemented.*[8, p. 45]"

Android applications are developed with Java. Java is the worlds most widely used programming language. It was chosen for Android development because it is powerful, free and open source. Java code is able to run on a variety of devices without any platform specific code.[8, p. 45]

2.3.1.3 PhoneGap for cross-platform development

"*PhoneGap is an open source framework for quickly building cross-platform mobile apps using HTML5, JavaScript and CSS. Building applications for each device–iPhone, Android, Windows Mobile and more– requires different frameworks and languages. PhoneGap solves this by using standards-based web technologies to bridge web applications and mobile devices. Since PhoneGap apps are standards compliant, they're future-proofed to work with browsers as they evolve.*[9]"

PhoneGap supports all of the latest iPhone and Android features including the Accelerometer, Camera and Geolcation, which make it a very appealing choice when beginning to develop an app.[10]

Using PhoneGap for cross-platform development isn't always an effective approach for a number of reasons, but primarily because the iPhone lacks a physical 'back' button. On the other hand most android devices have one built into their design. This means that certain pages would have to be designed differently for iPhone, as a virtual 'back' button would need to be included on the display.[11]

2.3.1.4 Mobile Operating System Selection

iOS will not be considered for this project due to the unfamiliarity of objective C, Swift and Xcode. Learning two new languages while also getting familiar with a new integrated development environment would have been nigh on impossible given the allocated for this project.

PhoneGap was definitely considered for this application as it caters for multiple mobile operating systems including both Android and iOS. However because it is developed using web-based languages it is not a preferable option for this project.

Android is the chosen mobile operating system for this project as it is developed primarily using Java, which has been practiced during the 3rd year of study in this computer science degree making it the preferred choice.

2.3.2 Integrated Development Environment (IDE)

In order to develop an Android application the Android SDK must be downloaded and linked to an IDE, where the coding will be done.

2.3.2.1 Eclipse

Eclipse was Android's IDE of choice for developing Android applications until December 2014. Some advantages of Eclipse are as follows:

- Code completion
- Refactoring
- Syntax checking

2.3.2.2 Android Studio

Android Studio was developed specifically for developing Android applications; it was stably released in December 2014. Once the stable version of Android studio was released, Google made it their official Android IDE.

2.3.2.3 IDE Selection

It made perfect sense to use the official Android IDE. Up until December 2014 Eclipse was used and once the stable release of Android studio happened, the project was migrated to Android Studio.

2.3.3 Expert Systems

Artificial Intelligence is the study and construction of agent programs that perform rationally in a given environment or agent architecture. It is a branch of computer science that is concerned with the automation of rational behaviour.

2.3.3.1 Drools

"Drools is a Business Rules Management System (BRMS) solution. It provides a core Business Rules Engine (BRE), a web authoring and rules management application (Drools Workbench) and an Eclipse IDE plugin for core development.[12]"

2.3.3.2 Prolog

"*Prolog is a programming language for symbolic, non-numeric computation. It is specially well suited for solving problems that involve objects and relations between objects.[13, p. 3]*"

2.3.3.3 Jess

"The Java expert system shell (Jess) is a rule engine and scripting environment written entirely in Oracle's Java language.[14]" In the past it hasn't been possible to implement Jess into an android application because the android platform is missing the entire Java beans package which Jess uses. However the newest version -Jess 8(still in its development phase) includes many bug fixes and comes with support for building Jess applications for Android devices.[15]

2.3.3.4 Expert System Selection

Prolog was written in C, which is very difficult to implement into an android application, therefore it wont be considered for this project.

Since Drools is designed specifically for a business management system, it may not be best suited to the proposed system and therefore wont be considered for this project.

Jess 8 is the most appealing option here. Since it was written in the Java programming language it is easily implemented into Java. Unfortunately Jess 8 was still in development during the initial planning of this project. After a number of communications with Ernest Friedman-Hill – (the creator of Jess) the conclusion was that it wasn't going to be released in time for implementation into this project.

As a result no expert system will be used in this project but instead an algorithm will be created in Java.

2.3.4 Databases

"A database is a data structure that stores organized information. Most databases contain multiple tables, which may each include several different fields.[16]" There are my

different types of databases available today. Examples of some of these are; Relational databases, Graph databases, No-SQL databases and Object-orientated databases.

The most popular type of database today is a relational database.[17] *"A relational database consists relations that are appropriately structured. We refer to this appropriateness as normalization.[18, p. 146]"* The relations are structured using tables that interact with one another using relationships. For example a Database may contain an employee and hours table. The 'employee' table would most likely contain the employees' personal details. Each employee may have an 'hours' table associated with them and this association is the relationship between the tables.

2.3.4.1 Considered Databases

SQL Lite
"SQLite is an amazing library that gets embedded inside the application that makes use of. As a self-contained, file-based database, SQLite offers an amazing set of tools to handle all sorts of data with much less constraint and ease compared to hosted, process based (server) relational databases.[19]"

PostgreSQL
"Compared to other RDBMSs, PostgreSQL differs itself with its support for highly required and integral object-oriented and/or relational database functionality, such as the complete support for reliable transactions, i.e. Atomicity, Consistency, Isolation, Durability (ACID).[19]"

MySQL
"MySQL is the most popular one of all the large-scale database servers. It is a feature rich, open-source product that powers a lot of web sites and applications online. Getting started with MySQL is relatively easy and developers have access to a massive array of information regarding the database on the internet.[20]" There are a lot of web-hosting websites that offer a free MySQL database that comes with a cPanel (control panel) for easy management of data.

2.3.4.2 Database selection

SQL Lite is more orientated to a client side database management system as opposed to a server side one. It would not suit this application for user accounts, as they should be stored on the server side. As a result SQL Lite will not be considered for this project.

PostgreSQL can be an over-kill when executing simple tasks and therefore may perform less than its counterparts such as MySQL. PostgreSQL also lags behind in terms of popularity. For these reasons it is very difficult to come across hosts or service providers that offer PostegreSQL database.[19]

MySQL satisfies the requirements of this project. There are many websites that offer free hosting and these include a MySQL database that is available to use straight away.

2.3.4.3 Manipulating data in the database

In order to access the MySQL database and manipulate data PHP scripts will be used.

PHP

PHP is a widely used, open source scripting language. A PHP script is a popular way to access and manipulate MySQL databases. PHP files can also contain HTML, CSS, and JavaScript. It is executed on the server and the result is returned to the browser as plain HTML by default. However it is possible to format the output as JSON and XML. It is easier to learn and get used to PHP, therefore it will be used in this project to manipulate data in the MySQL database.

JSON (Java Script Object Notation)

"JSON is a syntax for passing around objects that contain name/value pairs, arrays and other objects.[21]" "On the server-side you can easily encode/decode your objects to/from JSON.[21]" JSON will be used in this project to return data from the database back to the application.

2.3.5 API's

2.3.5.1 Google maps API

The Google Maps API allows any developer to integrate a diverse map with numerous possibilities into their application. There are available alternatives to the Google maps API: 'Open street map' being just one. However Google are a well-known multi-national corporation that has spent millions perfecting their mapping applications.

In order to integrate the Google maps API into an android application the developer needs to obtain an API 'key' that will grant them access to the API. This is done in the developers console, where the developer must register their application's digital certificate, known as its SHA-1 fingerprint[22].

2.3.5.2 Google Places API

"The Google Places API allows you to query for place information on a variety of categories, such as: establishments, prominent points of interest, geographic locations, and more. You can search for places either by proximity or a text string. A Place Search returns a list of places along with summary information about each place; additional information is available via a Place Details query.[23]"

Google Places is a useful API allowing the user to search for places within a specified radius from their current location. There are a number of parameters, which must be included in order for a search query to be executed; latitude and longitude, defined distances and the users API key. There are a number of optional parameters that can be added to the search query in order to narrow the results down to something more specific. These optional parameters include; keywords, name, type, min/max price and open now.

2.4 Weather Research

“Weather is widely believed to influence people’s mood. For example, the majority of people think they feel happier on days with a lot of sunshine as compared to dark and rainy days. Although this association seems to be common sense.[24, p. 662]”

The weather causes humans to act differently. In General, if it’s raining outside then most people would rather stay indoors for the day. If it’s sunny, most people would rather be doing something outdoors. For this reason alone, it is a good idea to only recommend activities to the user that can be done indoors if it is raining and to only recommend activities that are outdoors if it is sunny

2.4.1 Weather Survey

A weather survey was conducted at the beginning of December 2014 with the intention of getting a small gauge of people’s reaction to the weather. It was conducted using Survey Monkey and consisted of 10 questions. There were over 100 responses to the survey however the free version of Survey Monkey only allows the first 100 to be analysed.

The final question on the survey asked the users:
Do you think that a mobile app that will recommend activities to you based on your interests, age, gender, location, time of day and the weather is something you would download? 87% of respondents answered yes.

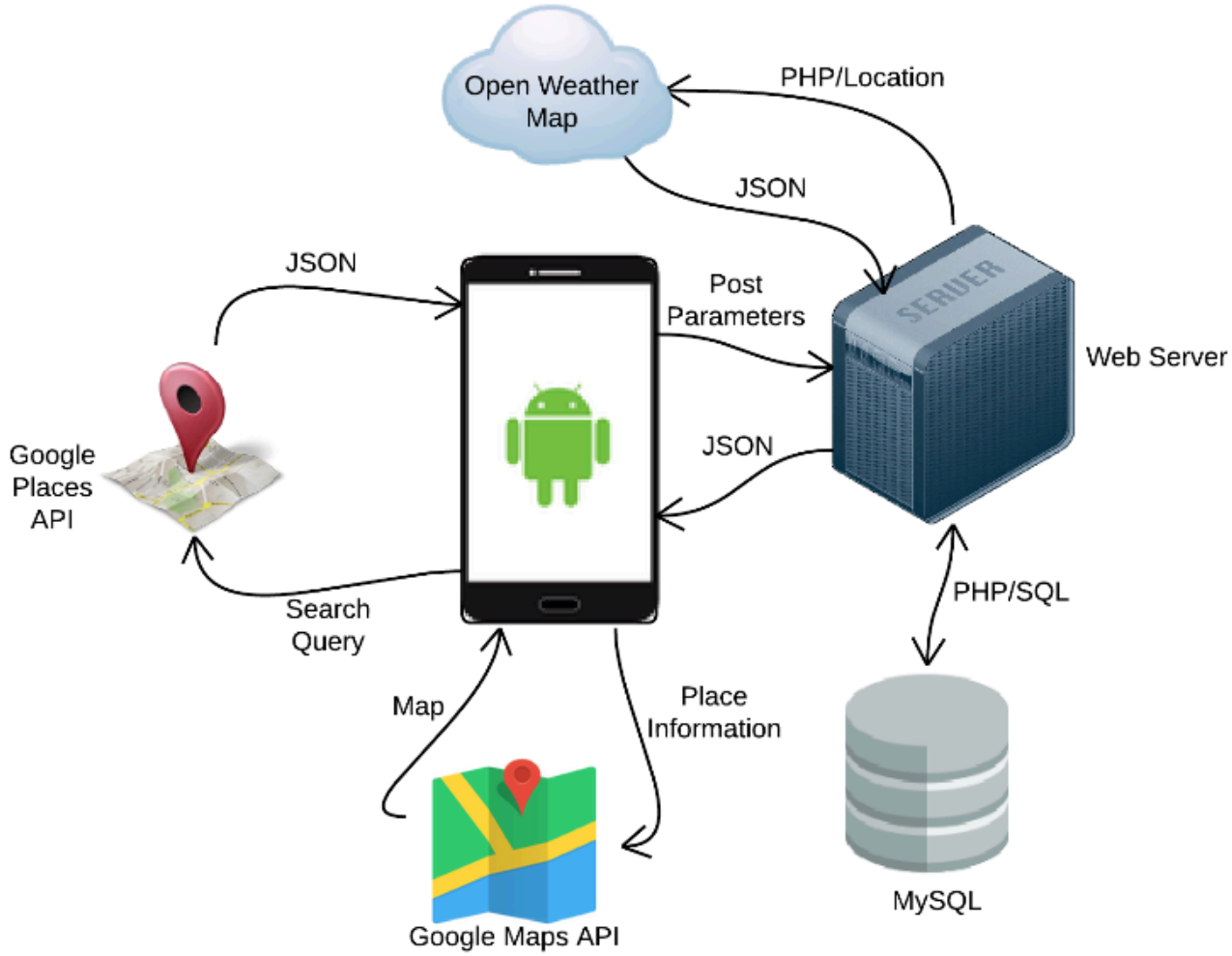
From the survey it is evident that the users mood is affected by the weather and that the weather alone has the power to change what they want to do on any given day.
The full survey can be viewed in Appendix A, located at the end of the document.

Chapter 3: Architecture and Design

3.1 Architecture & Development

3.1.1 Overview

Figure 5: Technical Architecture Overview



The technical architecture diagram illustrated in figure 5 demonstrates the technologies and devices that are used in this project.

- Java is the main language used in this project, it is used to control all of the functionality on the client side of the application.
- XML is used to create all of the layouts for the activities in this application including the button designs. However the majority of the fonts used are programmatically assigned using java.

- PHP is used to communicate with the MySQL database; scripts are stored on the server using the web host. The scripts are accessed using java.

- JSON is a format used to return results from the PHP scripts. These results could be data from the MySQL database, data generated in the PHP script or weather data from the Open Weather Map API.

- Google maps API is used to display a Google map on the android device, this is used so that we can show the user activities on a map.

- Google Places API is used to get nearby activities, the results are returned in JSON and the results can be displayed on the Google Map.

3.1.2 Identification Of Classes

The following code was written:

- LaunchScreen.java
- CreateAccActivity.java
- SignInActivity.java
- HomeScreen.java
- SearchSettings.java
- FullWeather.java
- Algorithm.java
- Account.java
- Connect.php
- CreateAcc.php
- SignIn.php
- FullWeather.php
- GetSettings.php
- PostSettings.php
- ChgPwd.php
- GetAct.php

The following code was sourced:

- CustomHTTPClient.java[25]
- ActivyJSONParser.java[26]
- ActivitiesActivity.java [26](Mostly sourced but also some unique code)

The following APIs were used:

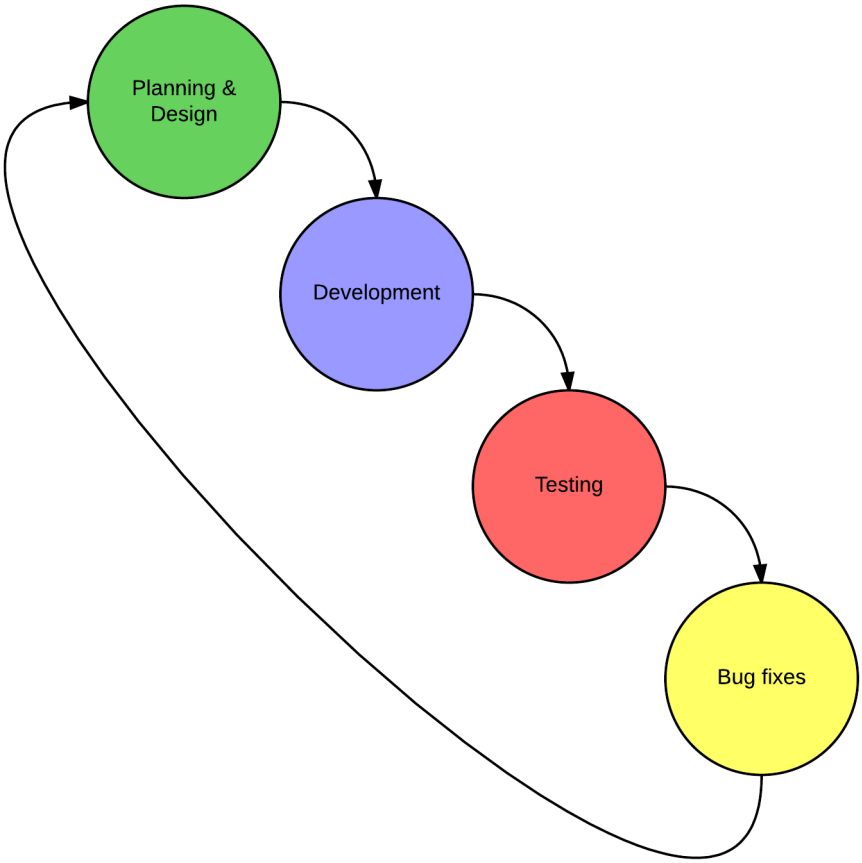
- Google Maps API
- Google Places API
- Open Weather Map API

The applications XML layouts were created entirely in Android Studio. The MySQL database was created with minimal coding. The SQL to create the tables and triggers was generated and submitted with the code.

3.2 Design Methodology

Choosing a software methodology at the beginning of any project is a very important decision, particularly so for large design teams. There are many pre- defined software methodologies available today such as agile, iterative, waterfall or rapid application development (RAD) software methodology. This project wasn't developed using any traditional software methodologies. The reason behind this is because it was an individual project being designed and implemented by one person. However there was a custom methodology in place, which is shown below in figure 6.

Figure 6: Software Methodology.



This methodology took both the agile development model and the iterative model and simplified them. It was chosen as the methodology for this project for a number of reasons:

- **Avoiding software bugs:** By planning and designing, developing, testing and fixing bugs throughout the project, there would be fewer bugs in the finished product. This is due to the fact that the project is being tested all the time and any problems that arise during this testing is solved immediately before looping back to the beginning and planning the next part of the project.

- **Avoiding design flaws:** Following this methodology would help prevent there

being any design flaws. If there was a design flaw identified towards the end of a project it could have a massive effect on the project reaching its deadline.

3.2.1 Step through of methodology:

1. **Planning & Design:** This is the first stage in the methodology; a plan is outlined for designing an aspect of the project. The plan may be for multiple parts of the project or just one single area.

2. **Development:** During this stage of the methodology the plan from the previous stage is implemented into the project via coding.

3. **Testing:** Once the development for something has finished it is then run to check if it works and if so, how well it works. A number of tasks are then performed to see how the application responds to them.

4. **Bug fixes:** Any problems that arise during the testing phase are troubleshot there and then, in order to try and come up with the best way to solve them or if need be work around them.

   After bug fixes the methodology begins again and the next part of the project is designed.

3.3 Use Case Diagrams

This section of the document illustrates the use cases for this project. The use case diagrams describe the different interactions the user will have from an end users point of view. These use case diagrams consist of an actor that is given the role of the user for all of the diagrams. The user will go down different paths depending on how they interact with the application. All of the diagrams show specific routes the user can take and the final diagram illustrates the complete application.

3.3.1 Registered User Sign In

Figure 7: Use Case Sign In

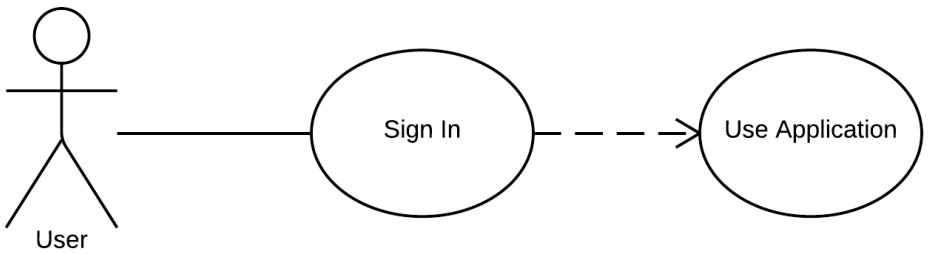


Figure 7 illustrates a high-level use case diagram that shows the flow of events when a registered user signs into the application. A user cannot use the application unless

they are signed into their account.

3.3.2 New User Create Account

Figure 8: Use Case Registration

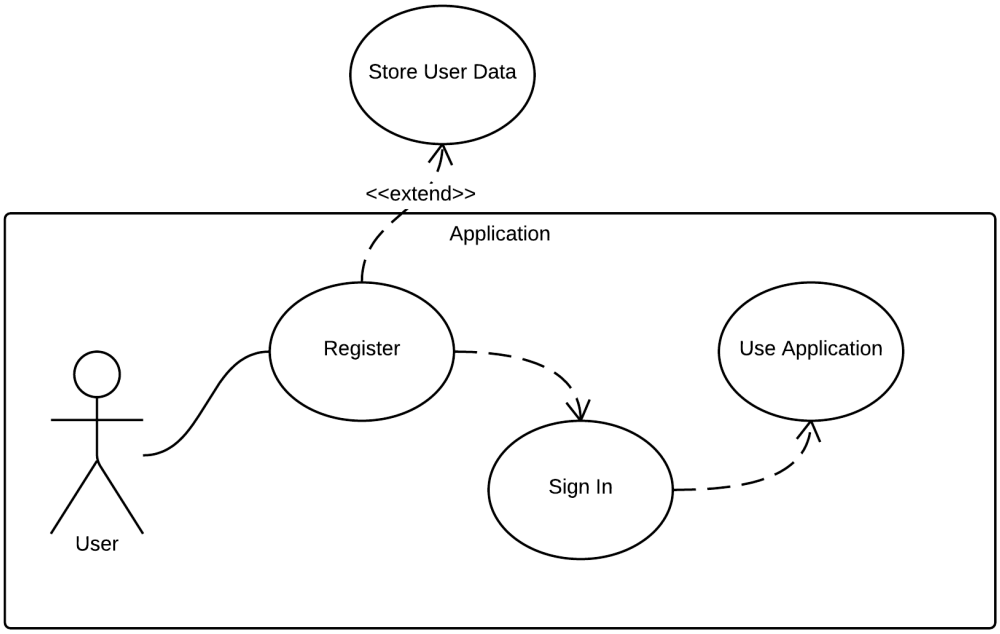


Figure 8 illustrates a high-level use case diagram that displays the flow of events when a user registers their details to the application for the first time. The user's details are stored externally and the user must sign in to use the application.

3.3.3 Home Screen Interaction

Figure 9: Use Case 'Home Screen' Interaction

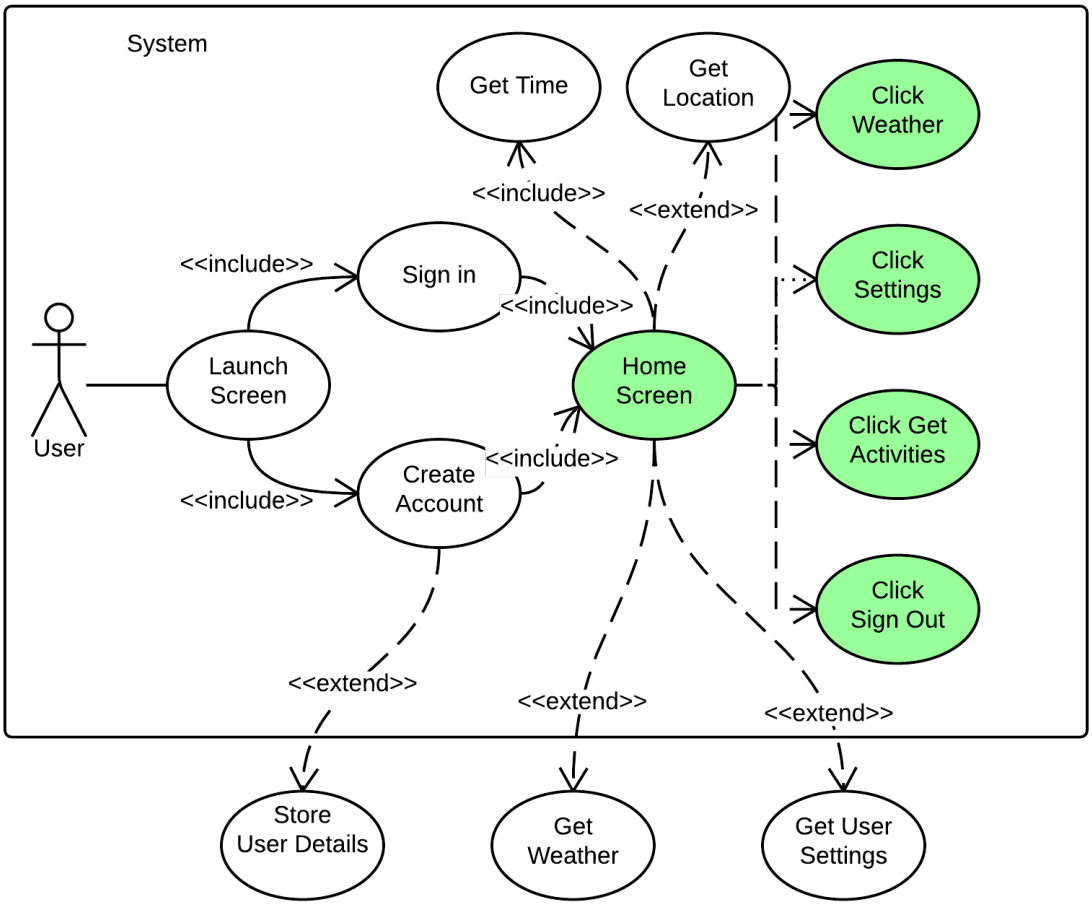


Figure 9 illustrates a high-level use case diagram that displays the flow of events when a user signs into the application and interacts with the home screen.

3.3.4 Weather Button Interaction

Figure 10: Use Case 'Weather' Button Interaction

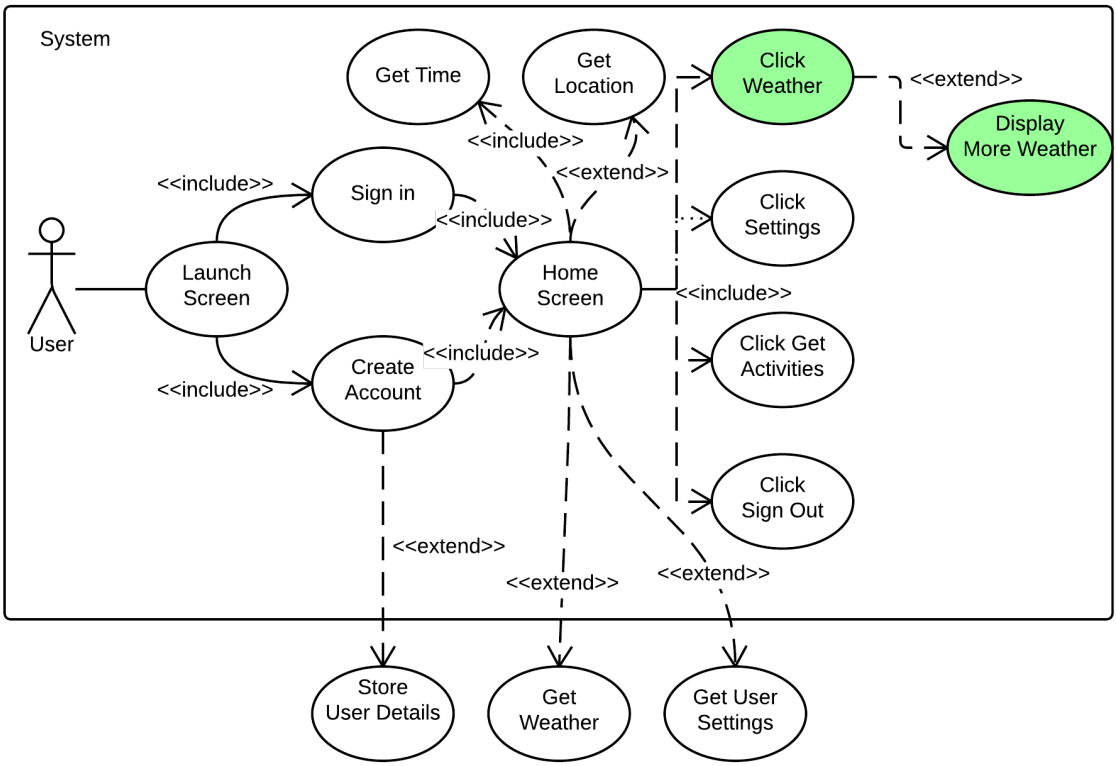


Figure 10 above illustrates a high-level use case diagram that displays the flow of events when a user signs into the application and interacts with the weather button on the home screen. More weather is displayed upon this click.

3.3.5 Settings Button Interaction

Figure 11: Use Case 'Settings' Button Interaction

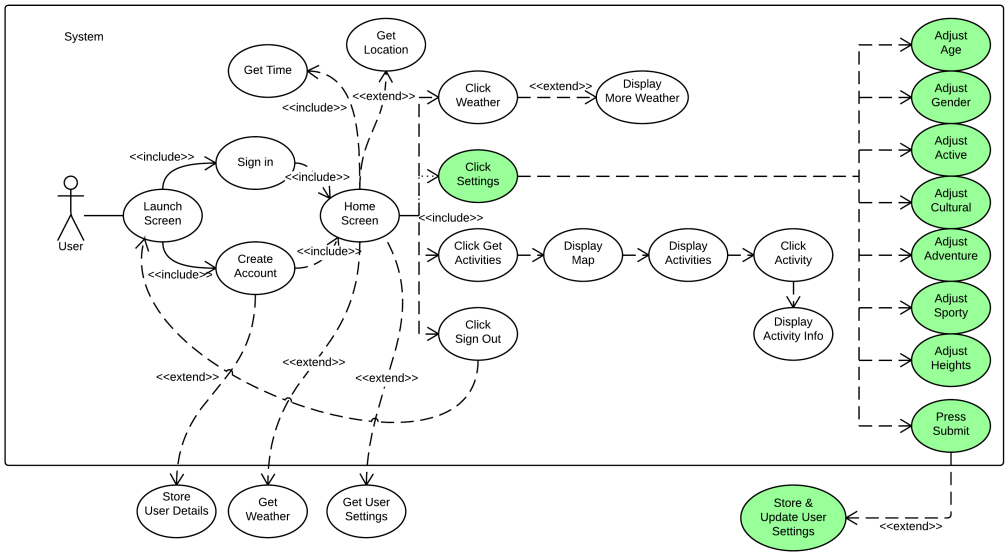


Figure 11 illustrates a high-level use case diagram that displays the flow of events when a user signs into the application and interacts with the settings button, now they are faced with a number of possible interactions. Once they press the submit button their changes are sent out of the application and updated.

3.3.6 Get Activities Button Interaction

Figure 12: Use Case 'Get Activities' Button Interaction

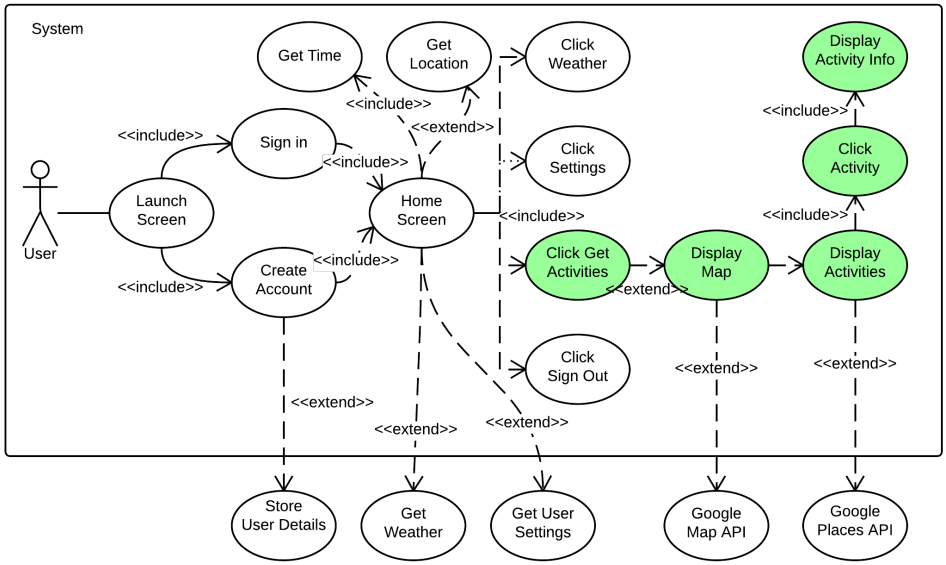


Figure 12 illustrates a high-level use case diagram that displays the flow of events when a user signs into the application and interacts with the 'get activities' button on the 'home screen'. The user is presented with a Google map

where nearby recommended activities are displayed. The user can interact with these activities by clicking on them where some information is displayed.

3.3.7 Full Application Use Case

Figure 13: Use Case 'Full Application'

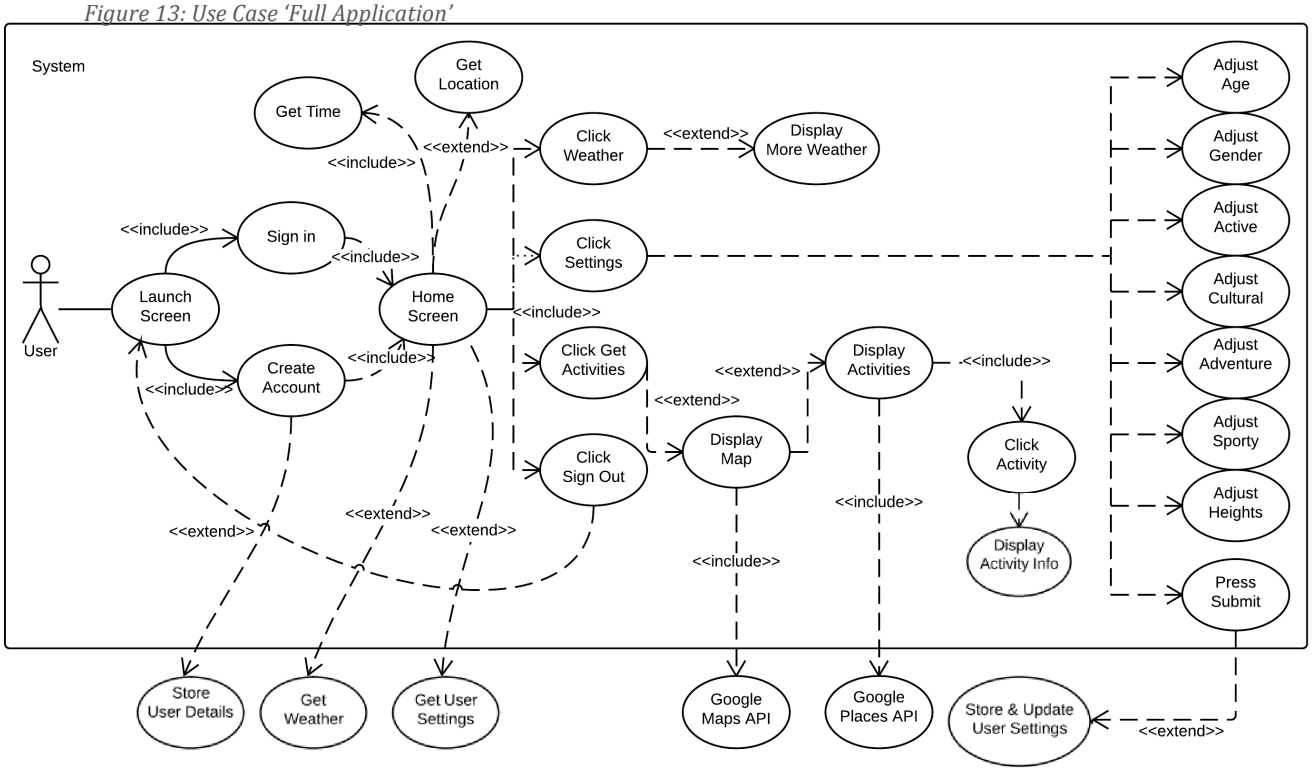


Figure 13 illustrates a high-level use case diagram that displays the complete flow of events that are possible when a user is using the application; everything here has been discussed in the previous diagrams.

3.4 Activity Diagrams

3.4.1 Create Account

Figure 14 Create Account Activity Diagram

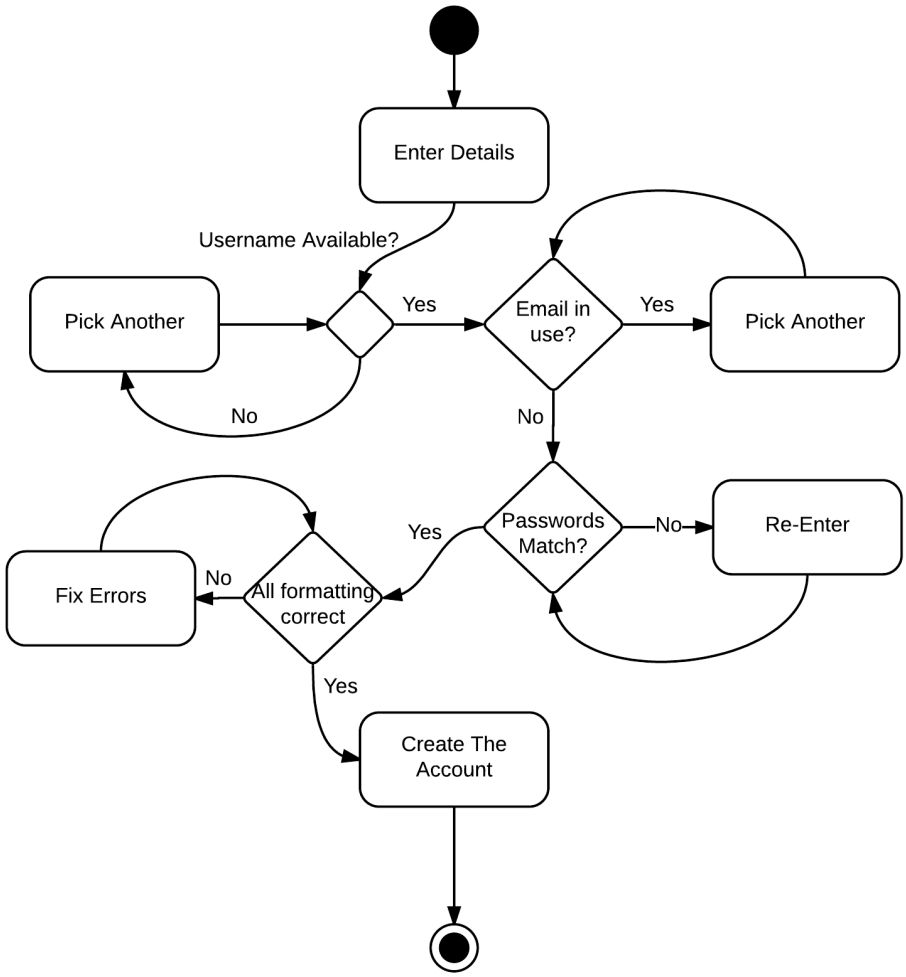


Figure 14 illustrates the process a new user goes through in order to create an account. The username and email address must be unique. Whatever is entered into these fields will be searched for in the MySQL database and if they exist the user will be presented with a message telling them so. The password entered must be entered twice in order to confirm it; if the passwords do not match then the user will be presented with an error. All of the fields must match a format specified, for example the username field must not contain any illegal characters and names should only contain letters and apostrophes. If any of the fields contain any illegal characters then the user will be presented with an error message. If all of the criteria are in order then the account will be created and the new user will be inserted into the database.

3.4.2 Sign In

Figure 15 User Sign in activity diagram

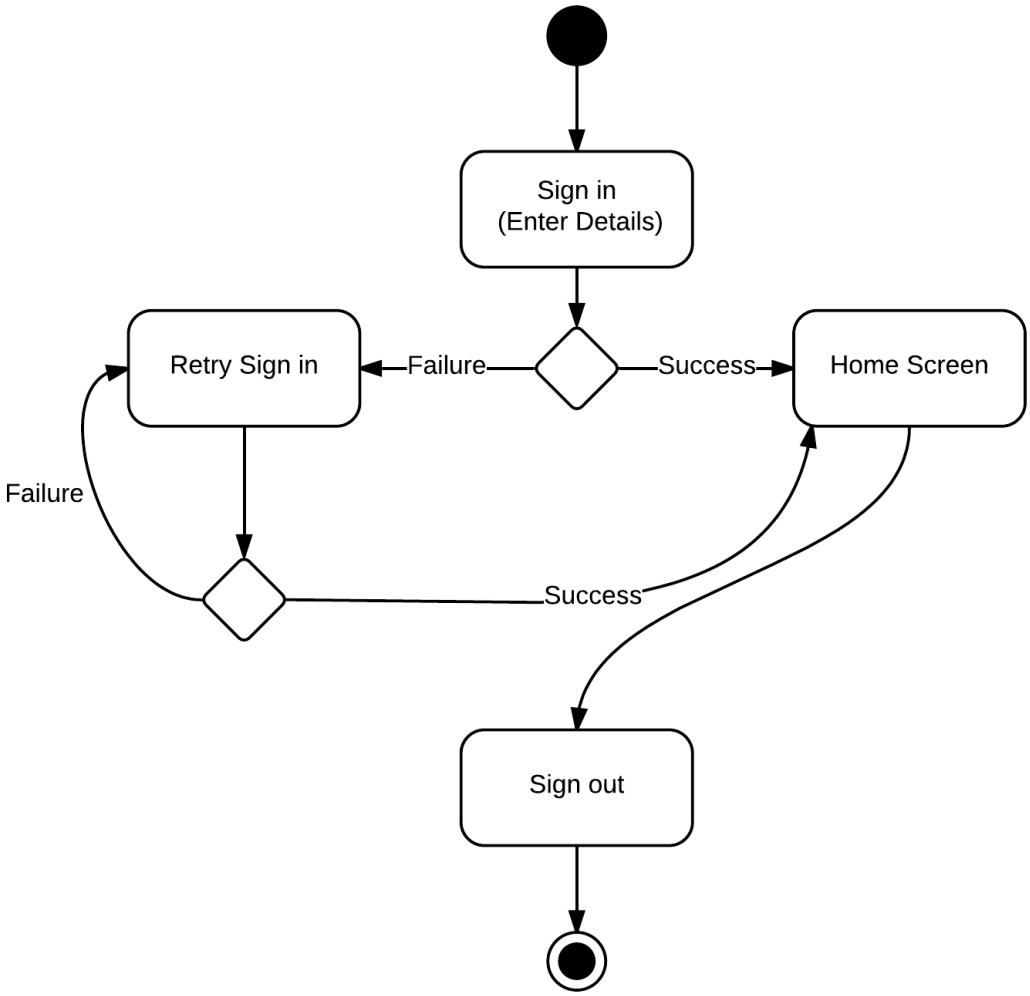


Figure 15 above shows the process that occurs when a user who is already registered attempts to sign in. When the user signs in, the application will query the database to check if the information provided is correct. If the user provides the correct details then they will be signed in successfully and brought straight to the home screen. If the users details do not match up they will be told their sign in details are incorrect and to try again.

3.4.3 Sign In/ Home Screen Simple

Figure 16 Simple Sign In and Home Screen Activity Diagram

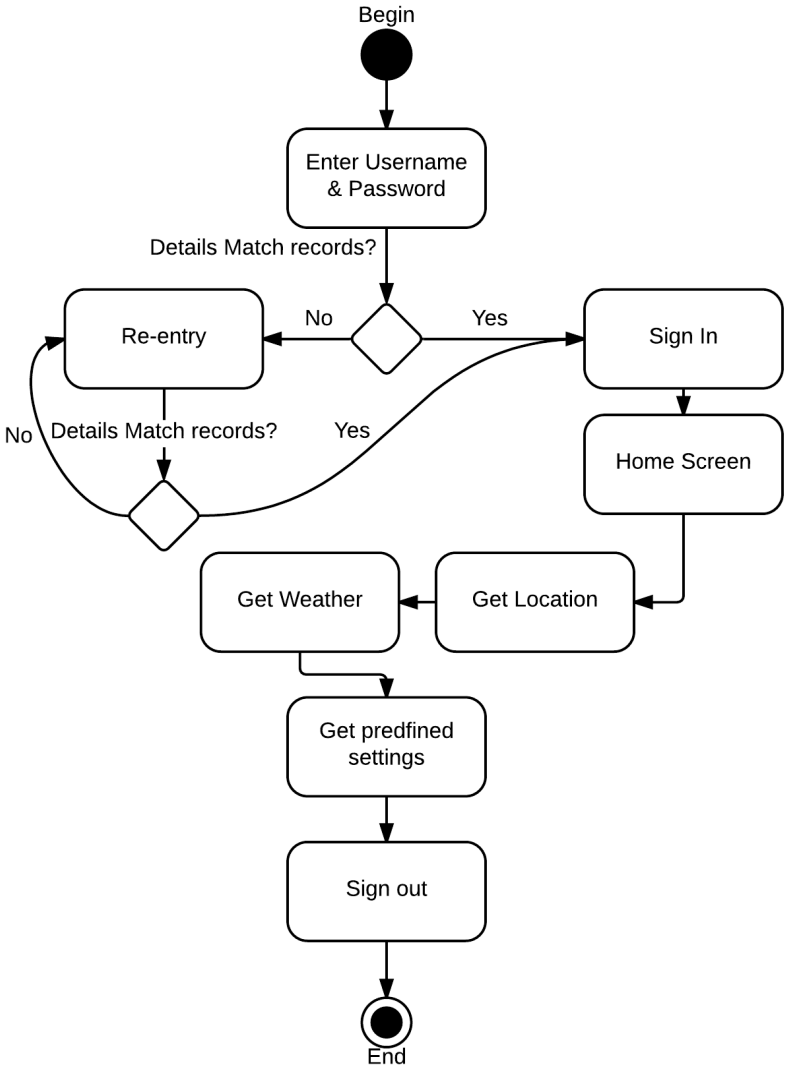


Figure 16 above shows the process in which a returning user will go through when signing back into the application. The user will be required to enter their username and password, which will then be checked, and if they match exactly with a record in the database they will proceed to the home screen or else they will have to try again. Once the user is logged in and the home screen is displayed, the following happens 1) The users latitude and longitude are stored in static doubles.2) The weather data is received and stored in static Strings. 3) The user's pre- defined settings are loaded from the database and stored in static integers and Strings. When the user presses sign out the home screen activity ends.

3.4.4 Home Screen

Figure 17 Home Screen Activity Diagram

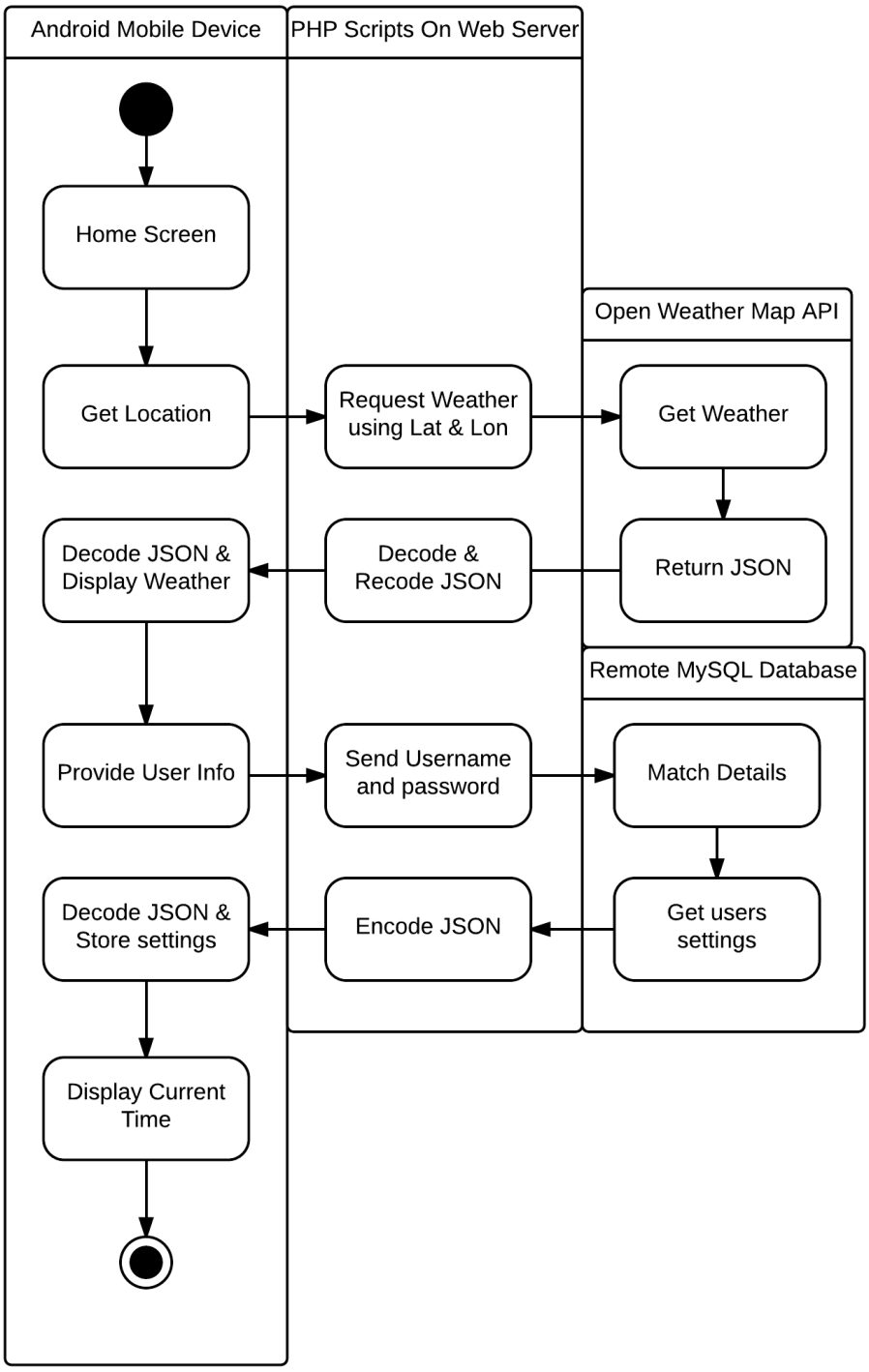


Figure 17 illustrates exactly what happens when the user accesses the Home Screen. The location of the user's mobile is sent to a PHP script in the web server. This in turn passes this location to an Open Weather map API call, which then returns the current weather in JSON. The PHP script decodes this JSON and then re-encodes some specific weather data to send back to the mobile. Once the

weather data is sent back to the mobile it is stored and displayed on the home screen.

The user's information is then sent to another PHP script and then straight to the MySQL database. If the users username matches a username stored in the database then the user's pre- defined settings are returned. They are encoded as JSON, decoded by the app and stored as static variables. The current time is displayed by the application.

3.4.5 Settings

Figure 18 Settings Screen Activity Diagram

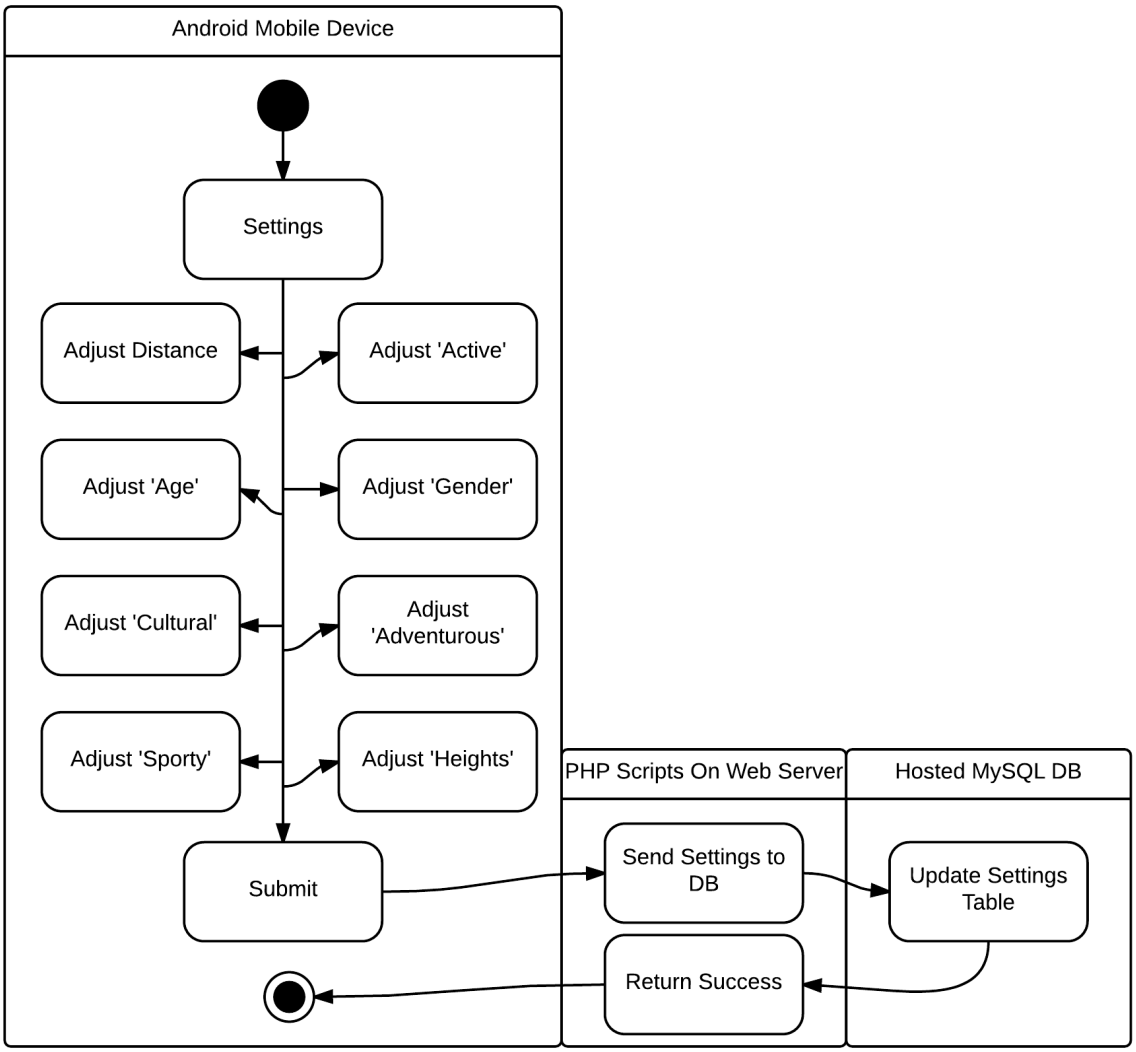
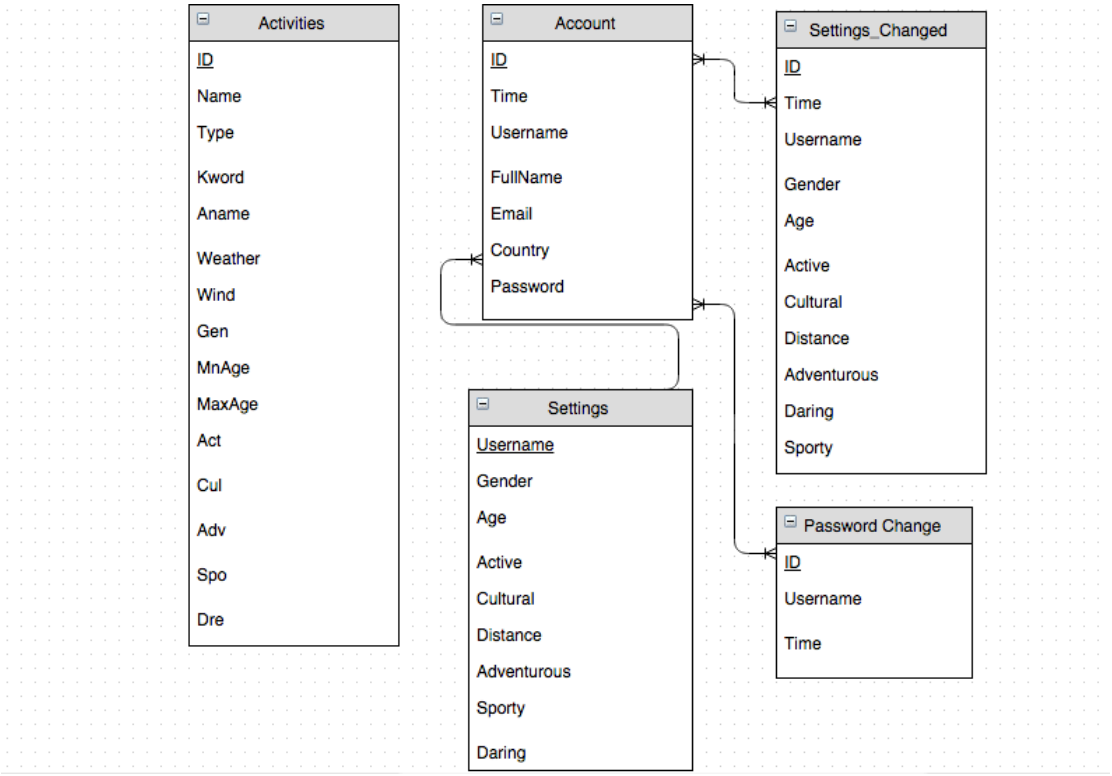


Figure 18 shows the 'settings' activity; here the user can adjust the different parameters in order to help with activity recommendation. When the user presses the submit button the new parameters are sent to a PHP script and then the 'settings' table in the MySQL database is updated. Once the update is complete a success message is sent back to the mobile and although not shown in Figure 18 once the success message is received the user is brought back to the home screen.

3.5 Database Design

Figure 19: MySQL Database



The database for this project didn't need to be large and complex so it was designed to be small and simplistic. Let's look at each table in the database from figure 19 in more detail:

Account:
This is where the users details are stored when they create an account. The primary key ID is auto incremented and a timestamp is inserted into the time field. The remainder of the fields are self-explanatory.

Settings:
The user's pre-defined settings are stored in here. Once the user creates an account, their username becomes the primary key of this table. When they log in for the first time, their settings will be set to default values. However when these are changed, an update is performed on the gender, age, active, cultural, distance, adventurous, sporty and daring, fields.

Activities:
This separate table is where the activities are stored. ID is the primary key and is auto incremented when a new activity is inserted. The name field contains the name of the activity; the text in this field is the only text from this table that will be visibly seen in the application. Google Places API's search parameters are stored in 'type', 'kword' and 'aname'. These fields contain words that will be used to help narrow down the search of an activity and display only those relevant.

The remainder of the fields in this table contain criteria that must be matched, in order for the activity to be returned.

Settings Changed:
This is a log table that records the user's changes in their search settings. After the settings table is updated, the user's username and new activities are inserted into this table using a trigger.

Password Change:
This is also a log table, it records every time the user changes their password.

3.6 User Interface design

When the first smartphone was released, user interface design became more important than ever. It is one of the most important parts of application development. The following steps were used to aid the User interface design:

- **Keep the interface simple.**

- **Create consistency and use common UI elements.** By using common elements in your UI, users feel more comfortable and are able to get things done more quickly.

- **Be purposeful in page layout.** Consider the spatial relationships between items on the page. Structure the page based on importance; careful placement of items can help draw attention to the most important pieces of information. It can also aid scanning and readability.

- **Strategically use colour and texture.** You can direct attention towards or away from items, using colour, light, contrast, and texture to your advantage.

- **Use typography to create hierarchy and clarity.** Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text help increase scan ability, legibility and readability.

- **Make sure that the system communicates what's happening.** Always inform your users of location, actions, changes in state, or errors.[27]

3.7 Algorithm Design

Figure 20: Algorithm Flowchart

Figure 20 illustrates the flowchart of the algorithm. The algorithm is expected to take input from eight variables. Weights are applied to the variables to help match activities. If the match is unsuccessful the weights are increased and the query is rerun. If it fails to match anything a second time the weights are increased once more for a final search. At this point, it will return at least one activity or it will return a fail.

For example, if the 'active' variable has a value of 50, initially the system will attempt to match any activities that have a value greater than or equal to 'active' - 20 and less than or equal to 'active' + 20. Therefore, any activities that have an 'active' value that lies between 30 and 70 are considered. However the criteria for all eight variables must match for the activity to be returned.

The weather and gender variables are matched in a different way. The system queries the database, trying to find a String that partly contains the string value of the variable. For example, if the weather column in the database contains the following String: " 'sun''clear''clouds' " and the column is queried for 'sun' it will match, as the String contains that value. The weather field contains a 0 and/or a 1 which both represent male and female. The search for gender is done the same way as the search for weather.

Table 1 shows the variable weights at each match attempt. Although it seems that the search range is very large, remember, every single variable has to match an activity's corresponding criteria in order for it to be returned.

Table 1: Algorithm – Matching Criteria Weights

Match Attempt	Variables	Weight
1	Weather	None
	Gender	None
	Age (Age)	Age >= MinAge && Age <= MaxAge
	Active (Act)	Act >= Act – 20 && Act <= Act + 20
	Cultural (Cul)	Cul >= Cul – 23 && Cul >= Cul +23
	Adventurous (Adv)	Adv >= Adv – 23 && Adv >= Adv +23
	Sporty (Spo)	Spo >= Spo – 27 && Spo >= Spo + 27
	Daring (Dre)	Dre >= Dre – 20 && Dre >= Dre + 20
2	Weather	None
	Gender	None
	Age (Age)	Age >= MinAge && Age <= MaxAge
	Active (Act)	Act >= Act – 30 && Act <= Act + 30
	Cultural (Cul)	Cul >= Cul – 33 && Cul >= Cul +33
	Adventurous (Adv)	Adv >= Adv – 33 && Adv >= Adv +33
	Sporty (Spo)	Spo >= Spo – 37 && Spo >= Spo + 37
	Daring (Dre)	Dre >= Dre – 30 && Dre >= Dre + 30
3	Weather	None
	Gender	None
	Age (Age)	Age >= MinAge && Age <= MaxAge

	Active (Act)	Act >= Act – 40 && Act <= Act + 40
	Cultural (Cul)	Cul >= Cul –45 && Cul >= Cul +45
	Adventurous (Adv)	Adv >= Adv – 45 && Adv >= Adv +45
	Sporty (Spo)	Spo >= Spo – 49 && Spo >= Spo + 49
	Daring (Dre)	Dre >= Dre – 42 && Dre >= Dre + 42

When the activity is returned to the application, a Google Place API request is created using the returned data. Then the string "opennow" will be appended to the request, to ensure only activities that are currently open will be searched for on the Map.

3.8 Prototyping

"A prototype is an initial version of a software system that is used to demonstrate concepts, try out design options, and find out more about the problem and its possible solutions.[28, p. 45]"

In this section of the document both the horizontal and vertical prototyping that were done for this project will be discussed.

Figure 21: Horizontal and Vertical Prototype[29]

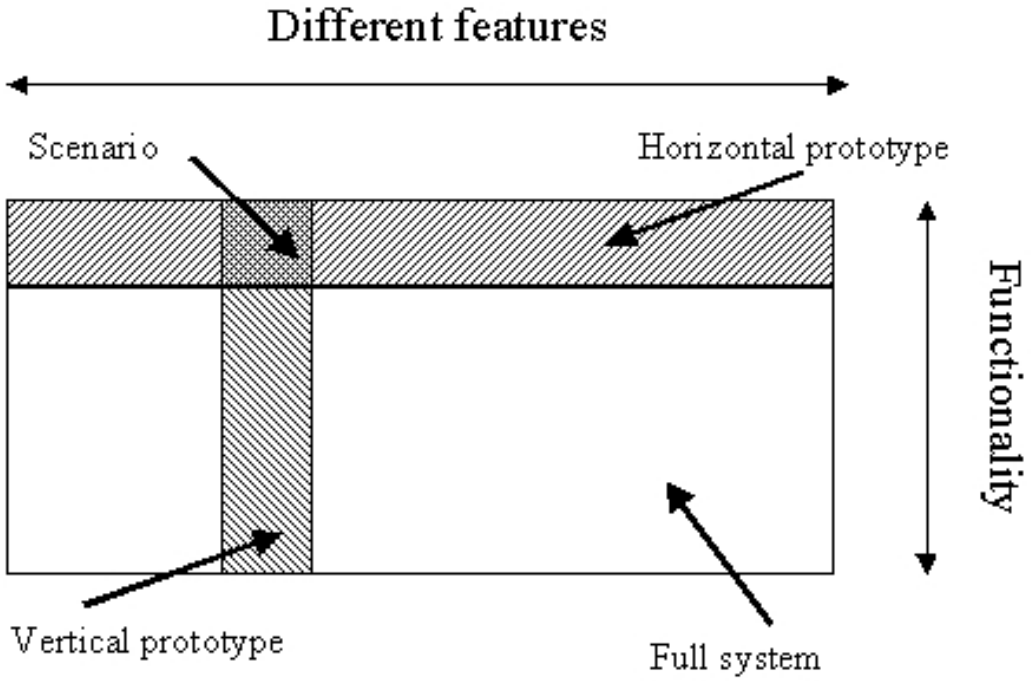


Figure 21 illustrates a system along with a horizontal prototype and a vertical prototype. The horizontal prototype models the entire system but with little or no functionality included while the vertical prototype models just part of the system but with complete functionality for that part.

3.8.1 Horizontal Prototyping

A horizontal prototype is a basic prototype that gives a broad overview of a certain part of an application or the whole application. There are 3 types of horizontal prototyping:

- Low fidelity: Paper prototypes where all of the screens the end user will interact with are drawn on paper
- Medium fidelity: A tool such as Lucid Chart is used to create the screens the end user will interact with.
- High fidelity: The prototypes actually work on a runnable device although they don't necessarily have to do anything.

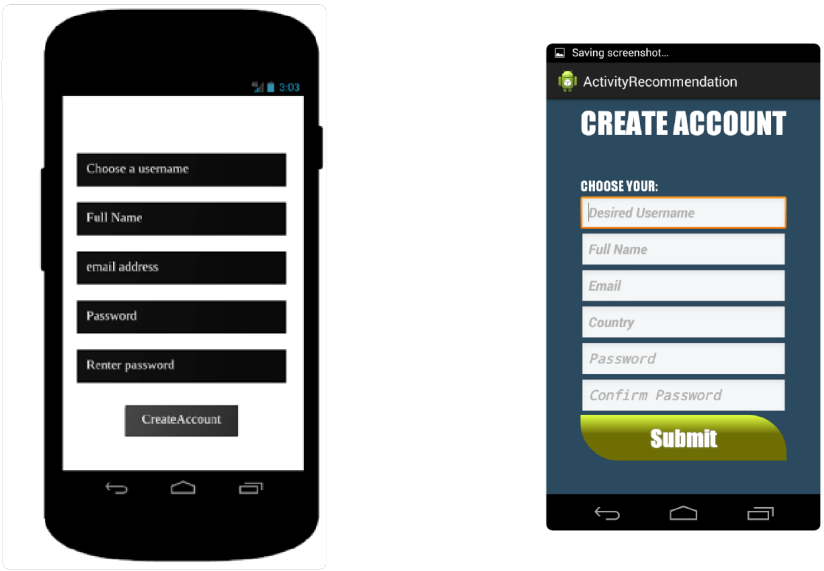
The majority of the prototyping for this project was done at a medium fidelity level. There was some low fidelity prototyping done at certain stages too. Refer to appendix D.

3.8.1.1 Create An Account Prototype

The 'create an account' screen is where the user will register their details with the system and allow them to sign in using their credentials. Figure 22 below shows the prototype for the 'create account' screen and the actual create account screen.

Figure 22: Create Account Prototype vs. Actual

Create Account Screen



Prototype
Late November 2014

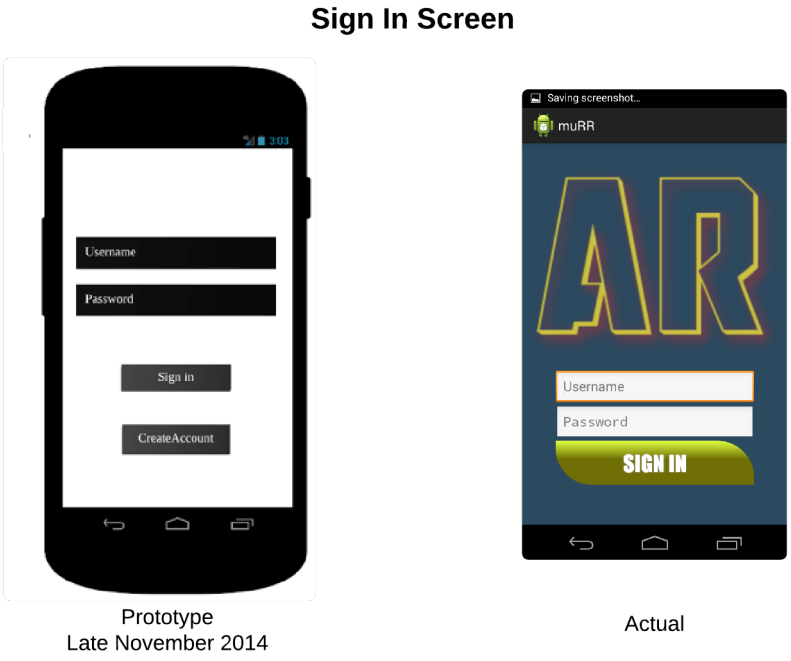
Actual

By looking at figure 22 it is evident that the actual 'create account' screen followed the layout of the original prototype. There are two slight changes to the actual 'create account' screen; there is a heading at the top of the page and an extra field that asks for the users country.

3.8.1.2 Sign In Prototype

The 'sign in' screen is where the user will enter their username and password to sign in and use the application. If the user does not have an account they will need to create one before this can be done. Figure TBC shows the medium fidelity prototype that was created in November 2014 beside the actual 'sign in' screen on the application.

Figure 23: Sign In Prototype vs. Actual



Prototype
Late November 2014

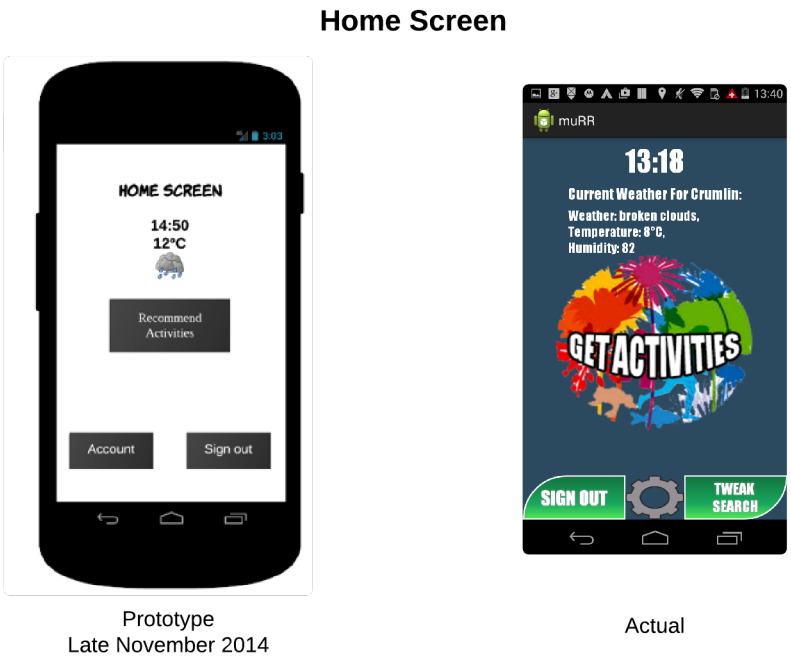
Actual

By looking at Figure 23 it is easy to identify that the 'create account' button illustrated on the prototype never made it into the actual application. This is because the application's 'launch screen' was used to display this option instead.

3.8.1.3 Home Screen Prototype

The home screen is what the user will see immediately after they sign into the application. On this screen they will see the current time and some weather details located up the top of the screen.

Figure 24: Home Screen Prototype vs. Actual



Prototype
Late November 2014

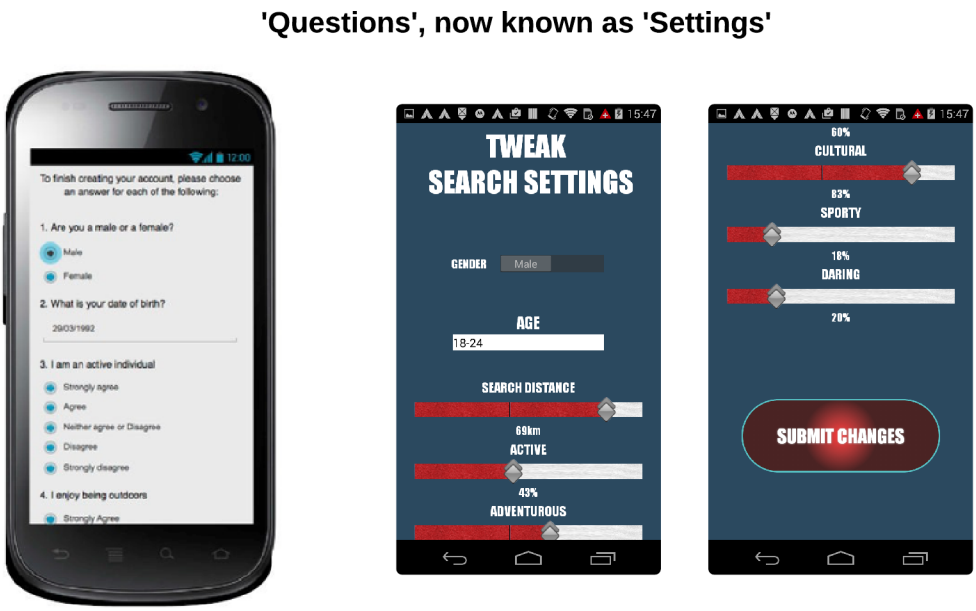
Actual

When the prototype of the home screen in figure 24 is compared to the actual home screen it is evident that the layout was followed. There are minor differences between the layouts of the two; the title 'Home Screen' isn't on the actual home screen and this was to make room for some extra weather information. The 'sign out' button is on the opposite side. Account is replaced with 'Tweak search settings'. In between these two buttons there is a cogwheel representing the account settings for the user that is currently signed into the application.

3.8.1.4 Questions Screen prototype

In the original prototype, after the user had created an account they would be brought to a screen called 'questions'. There they would answer a set of questions that would enable the application to learn a little bit about them. Although this screen was created in late January, it was changed soon afterwards when a better way to achieve the same results became apparent. A 'settings' page was created and instead of questions, key words were used. The user could adjust slider bars to indicate the level at which they wanted the keyword to contribute to the search. The user can easily adjust these answers at any time.

Figure 25: Questions/Tweak Settings Prototype vs. Actual



Prototype
Late November 2014

Actual

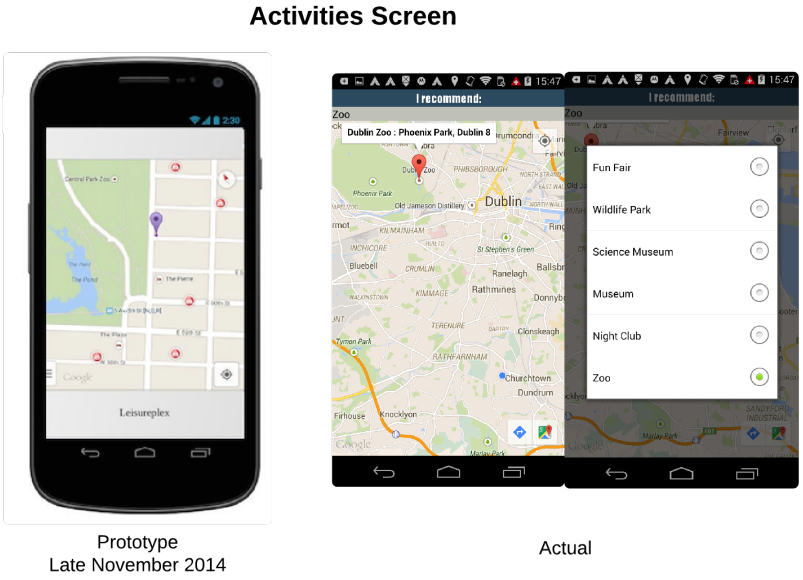
Looking at the prototype figure 25, there are a number of questions asked requiring the answer to be recorded using radio buttons. Looking at the actual implementation, 'gender' is recorded using a switch, 'age' is recorded using a spinner and the rest of the questions are recorded using seek bars. This was definitely a more efficient approach than the original idea.

The seek bars and use of single word headings as opposed to sentence long questions makes it much easier for the user to change these settings at any time quickly.

3.8.1.5 Activities Screen Prototype

The activities screen is where the recommended activities will be displayed. When the screen is first displayed the users current location is displayed as a blue blip. Red markers are displayed representing each activity. More information is given when each activity is clicked.

Figure 26: Activities Prototype



It is evident from figure 26 that the actual activities screen followed the original prototype. However there are a two small differences to be noted; the title of the activity is displayed on the top of the screen now and there is a dropdown list beneath it that displays some extra options that the user can select.

3.1.1.6 Extra screens

There were a number of extra screens that were added into the project despite never having any prototyping done for them. These screens are as follows:

- Launch Screen
- Account Settings Screen
- Extra Weather Screen

Launch Screen

The launch screen opens when a user opens the app. It is the first screen and acts like a welcome screen where a returning user can sign into their account and a new user can create an account.

Account Settings Screen

This screen allows the user to manage their account. Currently, the user is able to change their password here.

Extra Weather Screen

This screen allows the user to see a more detailed version of the weather information than what is displayed on the Home Screen.

3.8.2 Vertical Prototyping

"Vertical prototypes demonstrate the exact functionality of a product but for only a small section of the entire product. For example, a vertical prototype of a word processor might demonstrate all of the spell-checking functions, but none of the formatting or text-entry functions. All of the functions in a vertical prototype mimic their real counterparts as much as possible.[30]"

Vertical prototypes are most appropriate when a certain feature of a system is poorly understood and needs to be explored. For this project connecting to the remote database and manipulating data was definitely a less understood area of the project. To overcome this, a number of tutorials were followed in order to gain a knowledge and understanding of how remote databases are connected to and from mobile applications.

After learning from tutorials a sample application was created that connected to a remote database and allowed the user to input some data. This would generate a query that would then request some data from the remote MySQL database and then return the result as JSON. This JSON would be parsed and displayed to the user on the mobile device.

This sample application mirrored a lot of the necessary functionality for the proposed application. Once the process of connecting to a remote database and manipulating data was learned and understood, development on the create account and sign in screen could begin.

Chapter 4: Implementation

4.1 Introduction

How the application was developed will be discussed in this chapter. The methodology described earlier was a key factor during the implementation of this project, as the development was done in sections with testing afterwards. This allowed for optimal amounts of focus and concentration to be given to certain stages of the development.

4.2 Technology requirements

The following will be required to develop the proposed application:

- A Computer
- Android Studio
- Android SDK
- An Android mobile device
- Online hosting
- PHP scripts
- A MySQL database
- JSON
- Google maps API
- Google places API

4.3 Development Environment

In order to begin developing the application, a stable development environment was needed. The Eclipse integrated development environment (IDE) was already installed. However before development for android could begin the android software development kit (SDK) and the android development tools had to be installed and pointed to within eclipse. As discussed earlier in the document, once the Android Studio IDE was stably released the project would be migrated to that platform to continue development. Migration was incredibly easy, the android SDK had to be pointed to from within Android Studio and then Android studio was able to import the project and build it.

4.4 Server & remote MySQL Acquisition and set up

Once the project proposal was accepted it was understood that a server would be needed in order to host the MySQL database and PHP scripts that were going to be used to access it. After much online research an account was created on www.byethost.com where it was free to sign up and receive free hosting. It included a MySQL database and file manager where PHP scripts could be uploaded for execution. Unfortunately this particular webhost proved to be

unsuitable due to lack of features and a bad interface. A new account was created on www.000webhost.com. This new webhosting site offered the same features as the previous one along with even more. The vertical prototype as discussed earlier executed queries on a remote MySQL database that was hosted here. However after a couple of weeks an email was received stating that the free account had been removed due to a lack of unique visitors to the domain. As a result of the account being removed, paid hosting was reviewed and eventually an account on www.hosting24.com was created at €7 per month.

A custom domain name was created on www.hosting24.com; this allowed the creation of a domain that was easier to remember. The MySQL database was created along with the appropriate tables. The necessary PHP scripts were uploaded to the public part file manager, which would allow them to be accessed by the Android application.

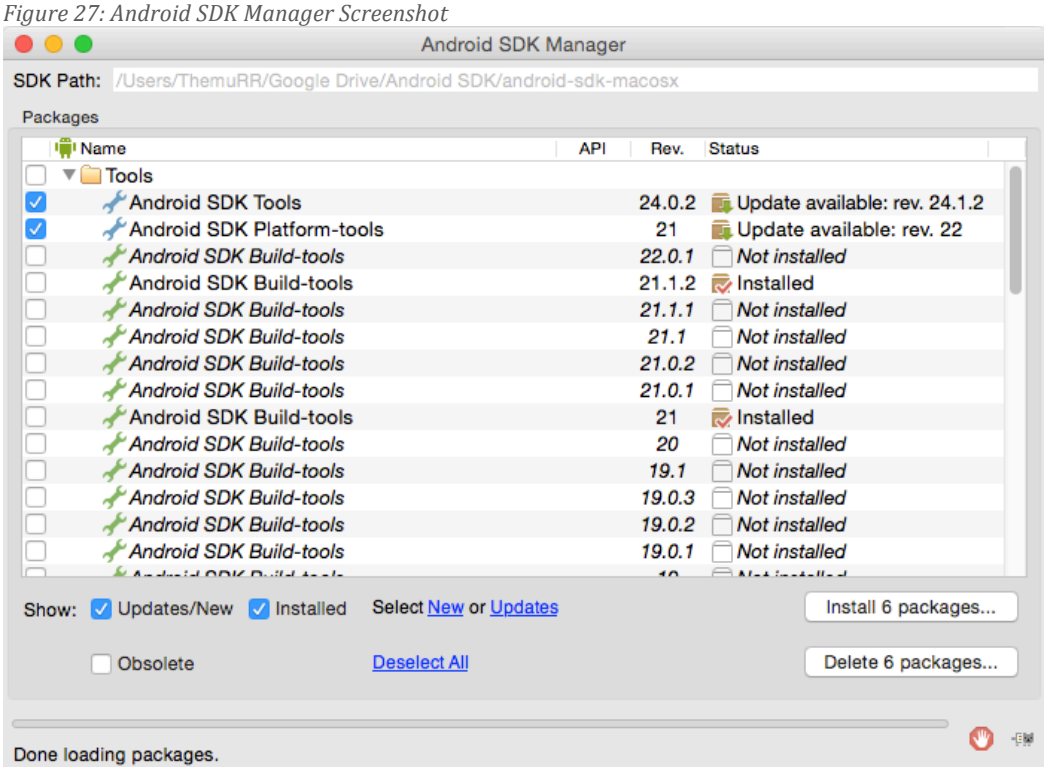
4.5 Development environment Setup

4.5.1 Android SDK Manager

The android SDK is compulsory to download and install when developing applications for Android devices. However once it was downloaded and pointed to from within the IDE more build tools and extras had to be installed. In order to install these build tools and extras, the Android SDK manager was opened from within Android Studio.

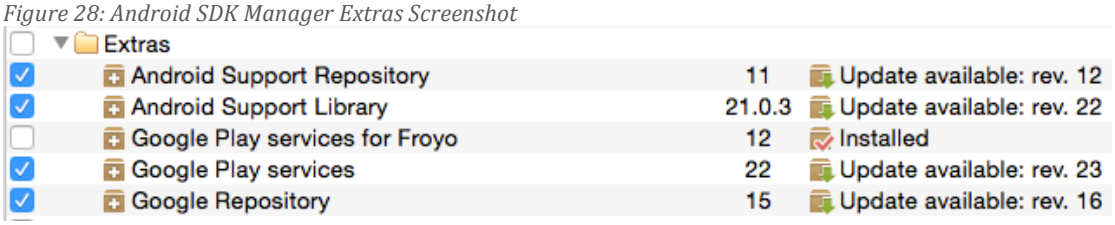
The Android SDK manager is full of build tools for every single version of Android. There are numerous support libraries, drivers and extras that include: the Google Play billing library and Google Plays services.

Figure 27: Android SDK Manager Screenshot



For this project there were a number of extras needed including the Google Play Services and the Android Support Library. These were installed by ticking their boxes and proceeding to click 'install X packages' located at the bottom of the SDK manager.

Figure 28: Android SDK Manager Extras Screenshot



4.5.2 IDE File organisation

Android Studio's file folders are very well organised, if the developer saves each file in the correct location it makes for a very efficient set up.

Figure 29: Typical Android Studio File structure Screenshot

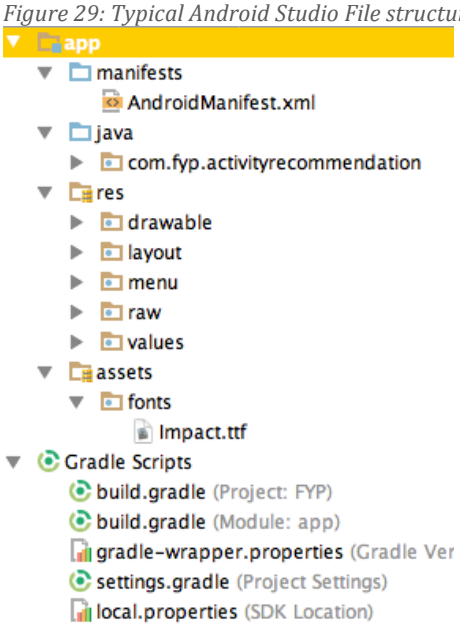


Figure 29 is a screenshot of the file structure in Android Studio for this Android application. There are two folders at the top level: app and Gradle Scripts. Inside the app folder there are four more folders titled: manifests, java, res and assets. Inside the Gradle Scripts folder there are five files titled: build.gradle, build.gradle, gradle-wrapper.properties, settings.gradle and local.properties.

Lets look at some of the folders in more detail:

Manifests
In the manifest folder the Android Manifest is stored. This manifest file presents essential information about the application to the android system. This file name must not be changed.

Java
In the Java folder packages can be found that store all of the java code developed for this project.

In figure 30 below the com.fyp.activityrecommendation package is expanded and we can see a number of java classes.

Figure 30: Java package expanded - Screenshot

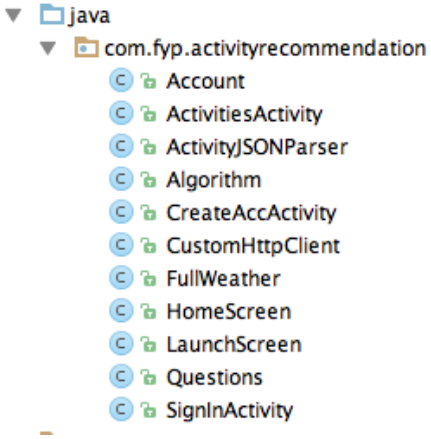
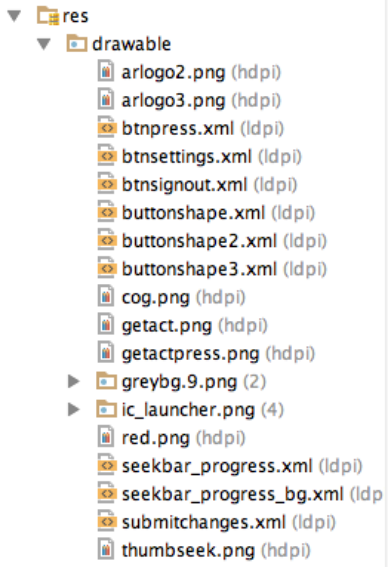


Figure 31: res/drawable - screenshot



Res

The resource (res) folder stores all of the applications resources such as layout files and images.

In figure 31 to the left we are able to see that the drawable folder contains a number of .png files and .xml files. The .png files represent the images seen in the application. The .xml files stored in this folder are instructions for the design of buttons and seek bars.

Figure 32: res/layout - screenshot

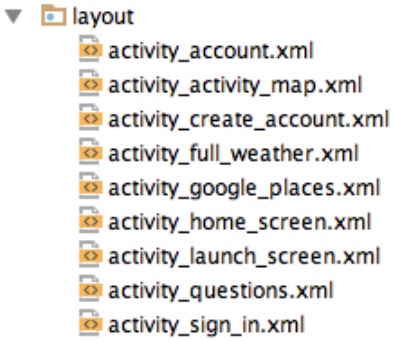


Figure 32 to the right shows the layout folder, which is also a sub folder of the res folder. In this folder all of the layout XML files are saved for the application.

Assets

After some research, an assets folder was added to the project to store any extra assets to the project. There is one fonts folder in the assets folder that is used to store any extra type fonts for the project.

4.6 Separate Development Environments

A separate environment was needed in order to code PHP. Since an Apple MacBook was being used for development, Notepad++ was not an option due to it not being compatible with the Mac OSX operating system. A simple interface similar to that of Notepad++ was a preferred. After potential candidates were reviewed, Sublime Text was chosen as the code editor for PHP scripts.

4.7 Application Development

4.7.1 Screen Layouts

The first thing that was implemented in this application was the majority of the screen layouts. The screen layouts themselves are obviously extremely important because without them there would be no application.

"A layout defines the visual structure for a user interface, such as the UI for an activity or app widget.[31]" In android development, layouts can be declared in two ways:

- "Declare UI elements in XML. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.[31]"
- "Instantiate layout elements at runtime. Your application can create View and ViewGroup objects (and manipulate their properties) programmatically.[31]"

All of the layouts for this project were created using XML. The use of XML layouts definitely suited this application, as it was easy to adapt to and learn XML. Android studio also allows the developer to drag and drop items in order to create a layout; this was a useful tool when unsure of the values that needed to be specified in order to place something such as a button in specific position.

To give an example of a layout used in the application, code snippet 1 shows the XML code for the home screen layout:

Code Snippet 1: Home Screen XML Layout

```
RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#ff2b495f"
   >


    <ImageView
       android:id="@+id/activityimg"


       android:background="@drawable/btnpress"
```

```
    android:layout_width="306dp"
    android:layout_height="251dp"
    android:layout_gravity="center_horizontal|top"
    android:layout_marginBottom="55dp"
    android:layout_above="@+id/signoutbtn"
    android:layout_centerHorizontal="true" />
<Button
    android:id="@+id/signoutbtn"

    android:text="SIGN OUT"
    android:textColor="#FFFFFF"
    android:textSize="25sp"

    android:layout_width="140dp"
    android:layout_height="60dp"
    android:background="@drawable/btnsignout"
    android:shadowColor="#A8A8A8"
    android:shadowDx="0"
    android:shadowDy="0"
    android:shadowRadius="5"
    android:layout_alignParentBottom="true"
    android:layout_alignParentStart="true"
    android:layout_toStartOf="@+id/cog" />
<Button
    android:id="@+id/settingsbtn"
    android:text="TWEAK &#xA;SEARCH"
    android:textColor="#FFFFFF"
    android:textSize="20sp"
    android:layout_width="160dp"
    android:layout_height="60dp"
    android:background="@drawable/btnsettings"
    android:shadowColor="#A8A8A8"
    android:shadowDx="0"
    android:shadowDy="0"
    android:shadowRadius="5"
    android:layout_alignTop="@+id/signoutbtn"
    android:layout_toEndOf="@+id/cog" />
 <RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_toStartOf="@+id/settingsbtn"
    android:layout_alignParentEnd="true"
    android:layout_below="@+id/textView5"
    android:layout_alignBottom="@+id/activityimg">
```

```
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Weather:"
            android:id="@+id/desc"
            android:textStyle="bold"
            android:textColor="#ffffffff"
            android:layout_below="@+id/area"
            android:layout_alignStart="@+id/area" />
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Weather Details for:"
            android:id="@+id/area"
            android:textColor="#ffffffff"
            android:textStyle="bold"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:textSize="20dp" />
    </RelativeLayout>
    <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textAppearance="?android:attr/textAppearanceLarge"
       android:text="TIME"
       android:id="@+id/textView5"
       android:layout_alignParentTop="true"
       android:layout_centerHorizontal="true"
       android:textSize="40dp"
       android:textStyle="bold"
       android:textColor="#ffffffff" />
    <ImageView
       android:id="@+id/cog"
       android:background="@drawable/cog"
       android:layout_width="80dp"
       android:layout_height="80dp"
       android:layout_alignTop="@+id/settingsbtn"
       android:layout_alignStart="@+id/textView5"></ImageView>
</RelativeLayout>
```

The XML code shown in code snippet 1 is self-explanatory to anyone with a programming background. Similar XML code will be used to create the final layout for all of the activities in this application.

4.7.2 Create Account

After the IDE had been set up the first thing that was implemented into this project was the create account screen. When the layout was being created a very basic one was used initially. There were six fields added to the screen:

- Desired Username
- Full Name
- Email Address
- Country
- Password
- Confirm Password

Once these fields were added in the layout some checks had to be done to ensure they followed the correct format. The username can only contain letters, numbers, underscores and hyphens. The full name can only contain letters, a space and an apostrophe. The email address can contain letters, numbers, the '@' symbol and a dot and it has to follow a specific pattern. The password and confirm password field can only contain letters and numbers.

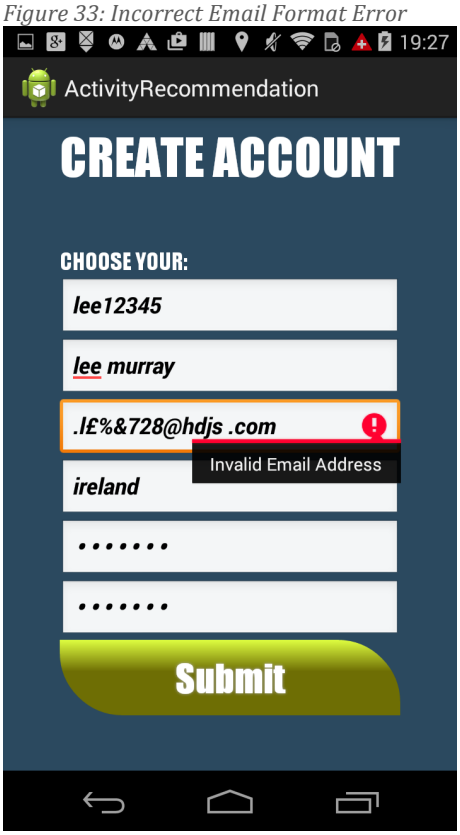
The following code snippet shows how the email format was coded:

Code Snippet 2: Email Pattern

```
public final Pattern EMAIL = Pattern.compile(
            "[a-zA-Z0-9+._%-+]{1,256}" +
            "@" +
            "[a-zA-Z0-9][a-zA-Z0-9-]{1,64}" +
            "(" +"."+"[a-zA-Z0-9][a-zA-Z0-9-]{1,25}" +")+");
```

If the user does not enter an email address that follows the format they will be presented with an error message as seen in figure 33. The previous code snippet and the error message seen in figure 33 are quite similar to the formatting code and error messages for the rest of the create account fields.

Figure 33: Incorrect Email Format Error



After the formatting is checked for all of the fields the system checks to see whether the two passwords entered match using the following code:

Code Snippet 3: Password matcher

```
if (password.getText().toString().equals(confirmpassword.getText().toString(
))) {
```

If the passwords match then the users inputted data is sent to a hosted PHP script using the following code:

Code Snippet 4: Http postParameters

```
// call executeHttpPost method passing necessary parameters
try {
        response = CustomHttpClient.executeHttpPost(
              "http://itsthemurr.com/createaccount.php", postParameters);
        String result = response.toString();
...
```

When the parameters reach the PHP script it then checks if the username or email exist in the database already. If the username is already in the database the script will return a '2' back to the application which will then present the user

with an error message similar to the one show previously in figure 33. Similarly if the email already exists in the database the script will return a '3' back to the application, which will present the user with an error.

If both the username and email address do not exist in the database then the account details are inserted into an account table in the database and a username is also inserted into another table called settings. A '1' is returned to the application to let it know that the account has been created successfully.

Here is the PHP script for creating an account:

Code Snippet 5: Create Account PHP Script

```
<?php

  include 'connect.php';

  mysqli_select_db($conn, $db_name);

  $user = $_POST["username"];
  $fullname = $_POST['fullname'];
  $email = $_POST['email'];
  $country = $_POST['country'];
  $pwd = $_POST['password'];
  $encpwd=md5($pwd);

  class result
  {
      public $value = "";
  }

  $sql1 = "SELECT username FROM account WHERE username = '$user'";
  $sql2 = "SELECT email FROM account WHERE email = '$email'";
  $outcome1 = mysqli_query($conn, $sql1);
  $outcome2 = mysqli_query($conn, $sql2);

  if (mysqli_num_rows($outcome1) > 0)
  {
      $result = new result();
      $result->value = "2";
  }
  else
  {
     if (mysqli_num_rows($outcome2) > 0)
     {
        $result = new result();
```

```
        $result->value = "3";
    }
    else
    {
        $sql = "INSERT INTO account (username, fullname, email, country, password)
                 values ('$user', '$fullname', '$email', '$country', '$encpwd')";
        $sql10 = "INSERT INTO settings (username) values ('$user')";
        if (mysqli_query($conn, $sql))
        {
             if (mysqli_query($conn, $sql10))
             {
                 $result = new result();
                 $result->value = "1";
             }
             else
             {
                $result = new result();
                $result->value = "4";
             }
         }
         else
         {
            $result = new result();
            $result->value = "4";
         }
    }
}
$output[] = $result;
print(json_encode($output));
$conn->close();
?>
```

4.7.3 Sign In

After create account was developed the next step was to develop a method that allowed the user to sign in to the application. A basic sign in layout was created and a username and password field were added along with a submit button. Unlike the create account page, the fields on the sign in page were not going to have any format checking. The reason behind this was it was believed to be better from a security point of view. Once the user enters their username and password they will press submit and then their username and password will be sent to a PHP script in the same way seen in the previous section. Then that PHP

script will check to see if that username exists in the database along with the password. If so the script will send back a '1' allowing the user to sign in or else it will send back a '2' telling the application that no credentials were found.

This is the code that tells the system what to do if they receive a '1' or a '2':

Code Snippet 6: Sign in, success/ fail

```
if (status == 1) {
  Toast.makeText(getApplicationContext(), "Success", Toast.LENGTH_LONG).show();

  Intent intent = new Intent(getApplicationContext(), HomeScreen.class);
  intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
  Intent.FLAG_ACTIVITY_CLEAR_TASK |
  Intent.FLAG_ACTIVITY_NEW_TASK);
  startActivity(intent);
  finish();

  username = username1.getText().toString();

} else if (status == 2) {
  Toast.makeText(getApplicationContext(), "Invalid Username or Password",
Toast.LENGTH_LONG).show();
}
```

If the system receives a '1' it will toast a message reading "success" and then it will launch the home screen. It will also clear the android device's back button history so that the user can not accidentally press the back button too many times and have to sign back in again. The username is also passed into a new static variable.

If the system receives a '2' it will toast a message reading "Invalid username or password", the system will not tell the user whether they got the username or password wrong as this is a common security practice in software development.

Because this application uses the Internet, almost every activity ensures that the user is connected to the Internet before it attempts to communicate with the database. This is done using the following code:

Code Snippet 7: Code for checking if the device is connected to the Internet

```
private boolean isNetworkAvailable()
{
   ConnectivityManager connectivityManager
       = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
```

```
   NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
   return activeNetworkInfo != null && activeNetworkInfo.isConnected();
}
. . .
if (isNetworkAvailable())
{ //Do Something}
else{ //Tell user they're not connected}
```

4.7.4 Home Screen activity

The home screen activity was the next part of this application to be developed. This activity would be where the user would see the weather information at the top of the screen along with the current time. There would be the option to get activities, visit account settings, tweak search settings and sign out.

The next step was to retrieve live weather information from the Internet. Since the end user could be anywhere in the world while using this mobile app, the weather would have to be retrieved via the Android device's latitude and longitude. This was achieved by taking advantage of the Android device's location manager.

Code Snippet 8: Location Manager

```
if (!isGPSEnabled())
{
   Toast.makeText(getApplicationContext(), "Turn on your location or the application
won't function properly",
   Toast.LENGTH_SHORT).show();
   Toast.makeText(getApplicationContext(), "Weather incorrect as your location is
off", Toast.LENGTH_SHORT).show();
}
else {

 LocationManager
 lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE); Location
 location = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
 if (location == null) {
  // request location update!!
   lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
   lon = location.getLongitude();
   lat = location.getLatitude(); }
 mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERV
 ICE);
```

```
location = mLocationManager.getLastKnownLocation(LocationManager.GPS_PROVID
ER);
if (location != null) {
  lat = location.getLatitude();
  lon = location.getLongitude();  }
else {
  mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
0, this);
  lat = location.getLatitude();
  lon = location.getLongitude(); }
```

A brief explanation of code snippet 8:
If the Android device's location settings are turned off, the user is presented with two warning messages about the application not functioning correctly. However if the device's location settings are turned on, then the device attempts to get the current location and store the latitude and longitude in 2 variables 'lat' and 'lon' that have previously been declared as static doubles.

The latitude and longitude are posted to a PHP script in a similar way to the method shown earlier in the document. Open Weather Map is used to retrieve the weather information, a query is sent from the PHP script to Open Weather Map, which then returns weather information in JSON. The desired weather information is decoded by the PHP script and then encoded back into JSON again. The reason the JSON was decoded and then encoded again was to return the weather information that was desired and in a preferable JSON format to that of Open Weather Maps.

The PHP code that retrieves the weather is as follows:

Code Snippet 9: Weather retrieving PHP script

```
<?php

$lat = $_POST["latitude"];
$lon = $_POST["longitude"];

$url="http://api.openweathermap.org/data/2.5/weather?lat=".$lat."&lon=".$lon."&units=
metric&cnt=7&lang=en";

// Make call with cURL
$session = curl_init($url);
curl_setopt($session, CURLOPT_RETURNTRANSFER,true);
$json = curl_exec($session);

$json=file_get_contents($url);

$data=json_decode($json,true);
```

```php
class weatherdata
{
                public $temp = "";
                public $weather = "";
                public $clouds = "";
                public $area = "";
                public $humidity = "";
                public $humidpressure = "";
                public $weatherdesc = "";
                public $tempmin = "";
                public $tempmax = "";
                public $windspeed = "";
                public $winddirec = "";
}

$data1 = $data['main']['temp'];
$data2 = $data['weather'][0]['main'];
$data3 = $data['clouds']['all'];
$data4 = $data['name'];
$data5 = $data['main']['humidity'];
$data6 = $data['main']['pressure'];
$data7 = $data['weather'][0]['description'];
$data8 = $data['main']['temp_min'];
$data9 = $data['main']['temp_max'];
$data10 = $data['wind']['speed'];
$data12 = $data['wind']['deg'];

$weatherdata = new weatherdata();
$weatherdata->temp = $data1;
$weatherdata->weather = $data2;
$weatherdata->clouds = $data3;
$weatherdata->area = $data4;
$weatherdata->humidity = $data5;
$weatherdata->humidpressure = $data6;
$weatherdata->weatherdesc = $data7;
$weatherdata->tempmin = $data8;
$weatherdata->tempmax = $data9;
$weatherdata->windspeed = $data10;
$weatherdata->winddirec = $data12;

$output[] = $weatherdata;
print(json_encode($output));
```

```
?>
```

A brief explanation of code snippet 9:
The latitude and longitude are stored in the variable '$lat' and '$lon' with are then concatenated to the '$url' string which contains the weather request URL. The URL is requested using cURL, which is executed in '$session'. The result is then passed into the '$Json' variable and is decoded and stored in '$data'. Further down the code, different weather parameters are stored in variables titled '$data1', '$data2' etc. Finally all of these variables are used to encode a new JSON array that then will be returned to the java code and parsed.

As stated previously, the latitude and longitude are sent to the PHP script (using 'post Parameters'), the returned weather data is then received and the JSON is parsed. This is done using the following code:

Code Snippet 10: Posting Latitude & Longitude, parsing JSON

```
try {
        response = CustomHttpClient.executeHttpPost(
        "http://itsthemurr.com/fullweather.php", postParameters);

        String result = response.toString();

        try {
        temp = "";
        weather = "";
        clouds = "";
        area = "";
        humidity = "";
        pressure = "";
        mintemp = "";
        maxtemp = "";
        wspeed = "";
        wdirec = "";
        wdesc = "";

        JSONArray jArray = new JSONArray(result);

        for (int i = 0; i < jArray.length(); i++) {
           JSONObject json_data = jArray.getJSONObject(i);
           Log.i("log_tag", "temp: " + json_data.getString("temp") +
            ", weather: " + json_data.getString("weather") +
            ", clouds: " + json_data.getString("clouds") +
            ", area: " + json_data.getString("area") +
```

```
            ", humidity: " + json_data.getString("humidity") +
            ", humidpressure: " + json_data.getString("humidpressure") +
            ", weatherdesc: " + json_data.getString("weatherdesc") +
            ", tempmin: " + json_data.getString("tempmin") +
            ", tempmax: " + json_data.getString("tempmax") +
            ", windspeed: " + json_data.getString("windspeed") +
            ", winddirec: " + json_data.getString("winddirec"));

            temp += json_data.getString("temp");
            weather += json_data.getString("weather");
            clouds += json_data.getString("clouds");
            area += json_data.getString("area");
            humidity += json_data.getString("humidity");
            pressure += json_data.getString("humidpressure");
            wdesc += json_data.getString("weatherdesc");
            mintemp += json_data.getString("tempmin");
            maxtemp += json_data.getString("tempmax");
            wspeed += json_data.getString("windspeed");
            wdirec += json_data.getString("winddirec");
        }
...
```

A brief explanation of code snippet 10:
The latitude and longitude are sent to www.itsthemurr.com/fullweather.php and the response is stored in the variable 'response' which is then passed into a new string called 'result'. A new JSON array is created using the contents of 'result' and then parsed using a 'for loop' and the results are stored as different strings.

Only some of the weather data returned is displayed on the home screen, if the user wishes to see more they can click on the weather text, which will bring them to an extra screen that displays more weather information.

The home screen activity is to display the current time to the user; in order to retrieve the current time the following code is used:

Code Snippet 11: Getting the current time

```
static Calendar c = Calendar.getInstance();
SimpleDateFormat dtime = new SimpleDateFormat("HH:mm");

String Time = dtime.format(c.getTime())
```

A brief explanation of code snippet 11:
The calendar function is used to retrieve the time, an instance of the calendar retrieved at runtime and stored in the variable 'c'. The desired format is set using

the SimpleDateFormat function and stored inside a variable called 'dtime'. A new variable 'Time' has its value set as the current time and format. This variable is then displayed on the screen using a TextEdit later on.

Next on the home screen the users 'search settings' are taken from the database and stored in variables. The application sends the user's username to a PHP script in the same way as shown previously. This PHP script will look for that username in the settings table and return all of the 'search settings'. However if it is a user's first time to use the application there will be no 'search settings' in the database. The script either returns the 'search settings' or else it gives an error. The java code will attempt to parse whatever the script returns and if it fails to parse the result, it assumes that there are no 'search settings' for the user. Therefore it treats them as a 1st time user, setting some default search settings and presenting them with a brief welcome message.

If the java code succeeds in parsing the result from the PHP script then the 'search settings' are stored in static variables. The code for parsing is rather similar to that of the code shown previously. The following code snippet shows what happens when there is an error parsing the 'search settings':

Code Snippet 12: Setting default search settings

```
catch (JSONException e) {
  Log.e("log_tag", "Error parsing data " + e.toString());
  Toast.makeText(getApplicationContext(), "Welcome! Head in to 'Tweak search
settings' to adjust what activities are recommended to you!",
Toast.LENGTH_LONG).show();
  gender = 0; // 0 is male, 1 is female
  age = 18;
  active = 50;
  cultural = 50;
  distance = 6; //6Kilometers
  adventurous = 50;
  sporty = 50;
  heights = 50;
}
```

A brief run through of code snippet 12:
If parsing the result fails then the 'catch' statement will be executed. "Error parsing data" is logged and the user is presented with a welcome message that read "Welcome! Head into 'Tweak Search Settings' to adjust what activities are recommended to you". Next, all of the 'search settings' variables are set to a default value that can later be changed by the user.

4.7.5 Tweak Search Settings Activity

This activity was initially a number of questions that were going to be asked to the user. Later the idea was modified to have: 1) A 'switch' to allow the user select their gender. 2) A 'spinner' to allow the user to select their age. 3) A series of 'seek bars' to allow the user to adjust different parameters, including the search distance. Part of the original idea to have a series of questions had been coded up but was later removed as the new idea came together.

The layout was created initially with the switch, spinner and seek bars in place and then some functionality was added to them. The 'search settings' variables were taken from the 'home screen' and stored in new variables. Then these variables were used to set the values of the switch, spinner and seek bars. This would ensure that a returning user would see the exact same settings they had set the last time they visited this activity, and also it would ensure that a new user was presented with the default settings.

Code Snippet 13: Setting values of Switch & Seekbar(s)

```
protected static int active = HomeScreen.getAct();
protected static int age = HomeScreen.getAge();
protected static int gender = HomeScreen.getGender();
protected static int cultural = HomeScreen.getCul();
protected static int adventurous = HomeScreen.getAdven();
protected static int sporty = HomeScreen.getSport();
protected static int heights = HomeScreen.getHeight();
protected static int distance = HomeScreen.getDistance()
...
 if (gender == 1){
  gendersw.setChecked(false); }
if (gender == 0){
  gendersw.setChecked(true); }
...
activeseek.setProgress(active);
distanceseek.setProgress(distance);
advenseek.setProgress(adventurous);
cultseek.setProgress(cultural);
sportseek.setProgress(sporty);
heightseek.setProgress(heights);
```

A brief explanation of code snippet 13:
Static integers are declared inheriting their values from the corresponding variable in the HomeScreen class. The switch is set to true or false depending on the value of the 'gender' integer. The seekbars values are set according to the value of their corresponding variables.

Once the user clicks 'submit changes' the updated settings are sent to the database in a similar way as the methods previously explained.

4.7.6 Account Settings Activity

This activity was added as an extra to allow the user to change their account password, as later on in the development it felt like this should be an option within the application. Three password fields were created, one for their current password and two for their new password. If the two new passwords match each other, the three of these values and the user's username are sent to a PHP script. The PHP script ensures that the current password exists and if so performs the update, changing the password. A record of this password change is recorded in the database using a trigger, which is shown in code snippet 14.

Code Snippet 14: Password Change SQL Trigger

```
CREATE TRIGGER `pwdchg` AFTER UPDATE ON `account`
 FOR EACH ROW INSERT INTO Password_Change VALUES ('id',NEW.username,DEFAULT)
//
DELIMITER ;
```

4.7.7 Activities Activity

This is where the activities are recommended and then displayed. The users information and settings are sent to a PHP script along with the weather. Activities are matched using the algorithm described earlier in the document. JSON containing the name for the activity is returned, along with a keyword, type and another name (to be used to find the activity). A loop is used parse this JSON and the results are stored in four separate string arrays. Figure 35 shows how the activities are stored in the database.

Figure 35: Database Activity Table Screenshot

Key_ID	name	type	kword	aname	weather	gen	minage	maxage	act	cul	adv	spo	dre
28	Amusement Park	amusement_park			'sun"clouds"clear'	'1"0'	0	55	40	25	65	35	75
25	Aquarium	aquarium			'rain"sun"clouds"clear"thunderstorm'	'1"0'	0	100	50	50	50	25	25
29	Art Gallery	art_gallery			'rain"clouds"clear'	'1"0'	20	100	30	70	60	20	20
30	Bar	bar			'rain"clouds"clear"snow"sun"thunderstorm'	'1"0'	18	65	20	40	20	20	20
2	Bingo		bingo		'sun"clear"clouds"extreme"thunderstorm'	'1"0'	50	100	20	30	20	20	0
20	Botanic Gardens		botanic+gardens		'sun"clear"clouds'	'1"0'	25	100	30	80	80	20	0
31	Bowling Alley	bowling_alley			'rain"clouds"snow'	'1"0'	0	40	60	30	40	60	20
32	Casino	casino			'rain"clouds"clear"snow"sun"thunderstorm'	'1"0'	18	100	30	20	40	20	20
35	Cinema	movie_theater	film		'rain"snow"thunderstorm"couds'	'1"0'	0	100	20	20	20	20	20
4	Crazy Golf		crazy+golf		'sun"clear"clouds'	'1"0'	0	100	50	20	60	50	0
10	Find a Tour		tour		'sun"clear"clouds"rain'	'1"0'	21	100	40	80	80	20	0
7	Fun Fair		funfair		'sun"clear"clouds'	'1"0'	0	45	40	60	70	20	50
11	Gaming Centre		gaming		'rain"extreme"thunderstorm'	'1"0'	0	35	20	20	20	20	0
14	Go Karting		go+karting		'sun"clear"clouds'	'1"0'	0	65	50	20	60	50	0
34	Gym	gym	gym		'rain"clouds"snow"clear'	'1"0'	16	55	90	20	30	80	20
17	Horse Riding		equestrian		'sun"clear"clouds'	'1"0'	0	100	50	40	80	60	30

The screen is displayed with a map fragment and a spinner on top. This spinner contains the names of the activities returned. The selected item on the spinner will be queried for using its corresponding keyword, type and name. This is how the query is constructed:

Code Snippet 15: Creating the places API search request

```
int index = selection.getSelectedItemPosition();
keyword = keys[index];
type = tys[index];
aname = anams[index];
...
StringBuilder request = new StringBuilder("https://maps.googleapis.com/maps/api/place
/nearbysearch/json?");
request.append("location=" + mLatitude + "," + mLongitude);
request.append("&radius=" + radius);
if (keyword != null)
    request.append("&keyword="+keyword);
if (type != null)
    request.append("&types="+type);
if (aname != null)
    request.append("&name="+aname);
request.append("&opennow");
request.append("&key=MyGoogleAPIkey");
```

A brief explanation of code snippet 15:
The spinner value is set using the array that contains the activity names. As a result of this, the position of the selected item on the spinner has a corresponding value in the same position in the keyword, type and 'aname' arrays.

The request is constructed using a string builder; this allows different strings to be appended to the request. The keyword, type and 'aname' Strings are appended to the request as long as they aren't null. Next "&opennow" is appended to the request; this ensures that all of the activities returned are currently open.

The request is sent and the results are returned as JSON, parsed and placed on the map. The activities are displayed using markers, if the user taps one of these markers then the name & vicinity is displayed. If no activities are returned the application advises the user to tweak the search settings and try again.

4.8 Implementing A Google Map Into The Application

The process of implementing a Google map into the application is more complex

than expected. Thankfully the instructions Google provided online were straightforward and easy to follow which simplified the process somewhat.

To begin the Google play services had to be added to the applications manifest, this was done with the following code:

Code Snippet 16: Google Play Services

```
<meta-data
   android:name="com.google.android.gms.version"
   android:value="@integer/google_play_services_version" />
```

Next an API key must be obtained.

4.8.1 Obtaining a Google API Key

"To access the Google Maps servers with the Maps API, you have to add a Maps API key to your application. The key is free, you can use it with any of your applications that call the Maps API, and it supports an unlimited number of users. You obtain a Maps API key from the Google APIs Console by providing your application's signing certificate and its package name. Add the key to your application by adding an element to your application's AndroidManifest.xml file.[32]"

Steps involved:

1. Retrieve information about your application's certificate.
2. Register a project in the Google APIs Console and add the Maps API as a service for the project.
3. Request one or more keys.
4. Add your key to your application and begin development.

The Google maps API key is based on a short form of the applications SHA-1 fingerprint. The fingerprint is unique so Google Maps uses it to identify applications that are making requests.[33]

There are two types of SHA-1 fingerprints; a debug fingerprint and a release fingerprint. The debug fingerprint is the one used when the application is still under development and the release fingerprint is the one used when the application is released to the Google Play Store.

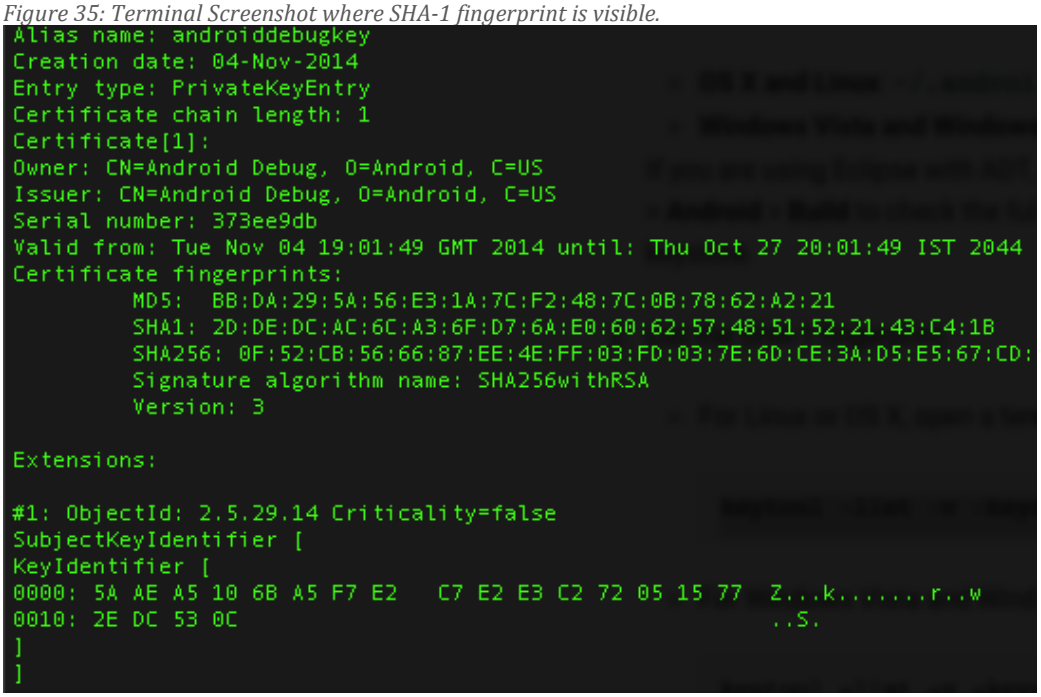
To get the SHA-1 fingerprint of this application the following code was entered into the Mac OSX terminal:

Code Snippet 17: Sha-1 fingerprint

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass
android
```

The above code produced the output seen in figure 34.

Figure 35: Terminal Screenshot where SHA-1 fingerprint is visible.



Once the project SHA-1 fingerprint has been obtained, an API project must be created in the Google developer's console and then an API key can be requested by entering in the SHA-1 fingerprint obtained earlier.

4.8.2 Adding the API key to the application

The Google API key is inserted into the Android manifest as follows:

Code Snippet 18: Manifest API Key

```
<meta-data
   android:name="com.google.android.geo.API_KEY"
   android:value="INSERT API KEY HERE"/>
```

4.8.3 Deploying The Application To The Google Play Store

- A Google Play developer's account was set up and a one-time fee of $25 was paid
- The application had to be signed digitally by generating a key store
- A launcher icon had to be created
- The Google API key had to be regenerated with a 'release' SHA-1 fingerprint
- Logging and debugging had to be turned off within the application
- Directories had to be reviewed and cleaned
- The manifest and Gradle build files had to be configured slightly.

After the steps above were completed the application was built for release and uploaded to the Google Play store as an APK file.

Chapter 5: System Validation

5.1 Testing And Feedback

Both black box and white box testing was carried out for this project. For black box testing a questionnaire was created and given to four users to complete while using the application.

For white box testing a series of test cases were created for important components within the project. They have a description, code segment and expected result column.

5.1.1 Black Box Testing

5.1.1.1 Questionnaire

There were 22 questions on the survey; the full results can be seen in appendix C. The questions were as follows:

1. Did the application launch successfully?
2. Did the "Create Account" button work?
3. Do you think the "Create Account" page is laid out well?
4. Could you create an account successfully?
5. Could you sign in?
6. Sign out of the application. Try to create another account but with the same username. Could you?
7. Sign back into the application but intentionally enter the wrong password. Could you sign in?
8. Sign in using the correct credentials. Did the weather information display correctly?
9. Was the location correct? If not please comment what it said and what it should be.
10. Click "Tweak Search Setting" if it is your first time doing so are the parameters set as follows: female, 18-24, 6km, 50, 50, 50, 50, 50?

11. Adjust all of the parameters and click submit. Did it work?

12. Go back into "tweak search settings", are the parameters the same as what you set them to?

13. Go back to the previous page and click the cogwheel at the bottom, attempt to change your password. Could you?

14. If your password changed, sign out and sign in with your old one. Did it work?

15. Sign in with your new password. Did it work?

16. Click on "Get Activities" did any activities get recommended? If not tweak the search settings and try again.

17. Click on the activity to view a dropdown menu, are there more activities? Please comment the activities that were recommended.

18. If so click on the new activity. Did anything display on the map?

19. Click on one of the markers. Did it display the name and vicinity?

20. Rate the design out of 10

21. Rate the application out of 10

22. Do you feel that there is any features missing that could be added?

Overall the test results were good and the user's feedback was positive. There were a couple of problems identified that were then worked on. The current location that Open Weather Map returns isn't always correct. This is known issue that lies with Open Weather Map. In the future the weather API could be changed to fix this.

5.1.2 White Box Testing

White Box Testing is testing a software systems internal functionality. Specifically how the code works not the overall system. It was necessary to include test cases for actions that were deemed importatnt. The results can be seen in Appendix B.

Test Cases

Test ID	*Description*	*Method/Code*	*Expected Result*
1.	App launches	Pressing the app launcher	Launch Screen Opens
En	User enters a	SELECT username FROM account WHERE	'2' is passed back to

	username	username = '$user'; if(mysqli_num_rows($outcome1) > 0) { $result = new result(); $result->value = "2"; }	the app if the username exists or '1' is passed back if it does not exist.
3.	User enters an email address	SELECT email FROM account WHERE email = '$email'; if (mysqli_num_rows($outcome2) > 0) { $result = new result(); $result->value = "3";}	'3' is passed back to the app if the email exists or '1' is passed back if it does not exist.

4.	User confirms their password	If (password.getText().toString().equals(confirmpassword.getText().toString()))
		{
		//Send details to DB
		}
			The passwords must match in order to proceed with account creation. There is an error if the passwords don't match.

5.	User submits their unique account details.	"INSERT INTO account (username, fullname, email, country, password) values ('$user', '$fullname', '$email', '$country', '$encpwd')";	The details entered by the user will be inserted into the MySQL DB.
6.	Sign in page launches after account created		User will receive a message stating 'Account successfully created' and be prompted to Sign in.
7.	The user signs into the application	"SELECT * FROM account WHERE username = '$user' AND Password = '$encpwd' LIMIT 1";	'1' is passed back to the app if both username and password match records in the DB. Otherwise '2' is passed back.
8.	The app gets the current location.	Using the Location Manager and the mobiles GPS	lat and lon are stored as static Doubles.
9.	The weather is displayed	The lat and lon are passed to a PHP script that returns current weather data in JSON, which is then parsed and the data is stored as static variables.	Weather is displayed at the top of the home screen.
10.	Clicking on the weather shows more weather	A new Intent begins and methods containing the weather data are called from the Home screen and stored in TextViews.	A new page opens displaying even more weather data.
11.	The current time is displayed.	Calendar c = Calendar.getInstance(); SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");	The current time is displayed on the home screen with

		String Time = sdf.format(c.getTime()); tv6.setText(Time);	the following format: (HH:MM)
12.	Users settings are received into the application.	SELECT gender,age,active,cultural, adventurous,outdoors,sporty,heights FROM settings WHERE username = '$user'	The users saved settings are displayed on the settings page. If it is their first time to use the app some pre-defined values are set.
13.	User clicks submit on the 'tweak search settings' page	"UPDATE settings SET gender = '$gender1',age = '$age1',active = '$active1', cultural = '$cultural1', adventurous = '$adventurous1', outdoors = '$outdoors1', sporty = '$sporty1', heights = '$heights1' WHERE username = '$user'";	The users presses submit from setting and the DB is updated with the new settings.

14.	Age spinner is set to the value of the users age if already defined	if (!compareValue.equals(null)){
		int spinnerPostion = dataAdapter.getPosition(compareValue);
		agespin.setSelection(spinnerPostion);
		spinnerPostion = 0; }
			The users age range will be set on the spinner the 2nd time they visit the search settings page

15.	Seekbars are set to value previously set by user	activeseek.setProgress(active); …	The value of the seek bar(s) is set to the same value of their corresponding variable.

16.	Switch is set to previously defined gender.	if (gender == 1)
		gendersw.setChecked(false);
		if (gender == 0)
		gendersw.setChecked(true);
			The value of the switch is set to either male or female.

17.	User Signs Out	intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |Intent.FLAG_ACTIVITY_CLEAR_TASK |Intent.FLAG_ACTIVITY_NEW_TASK);	The user can't press the back button and be brought to the home page once signed out

18.	The back button is disabled on the Home Screen	@Override
		public void onBackPressed() {
		// Do Nothing
		}
			The user will not accidentally sign out on the home screen if they keep pressing 'back'

19.	System checks if the GPS is enables	public boolean isGPSEnabled() LocationManager lm = (LocationManager)getSystemService(Context.LOCATION_SERVICE); return lm.isProviderEnabled(LocationManager.GPS	An if statement can be used to ensure the GPS of the device is enabled before performing a task.

		_PROVIDER); }	
20.	System ensures it is connected to the internet	private boolean isNetworkAvailable(){ ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE); NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo(); return activeNetworkInfo != null && activeNetworkInfo.isConnected();}	An if statement can be used to ensure the devide is connected to the internet before performing a task.

22.	Google Camera is looking at users location	mGoogleMap.setMyLocationEnabled(true);
		
		mGoogleMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
			When the map launches the users location will be the centre point

23.	Markers are added to the activities location	LatLng latLng = new LatLng(lat, lng); markerOptions.position(latLng);	Markers display at the Lat and Long of the activities

Chapter 6: Project Evaluation

6.1 Challenges and Learning Outcomes

In a project of this magnitude there are always going to be a number of challenges that arise during the development. Luckily the vast majority of them were overcome. Firstly, connecting to the remote MySQL database was difficult. Many tutorials were followed and watched on YouTube and it was eventually figured out after approximately two weeks. While overcoming this particular challenge it was learned that in order to connect to a remote database more than one programming language might have to be used.

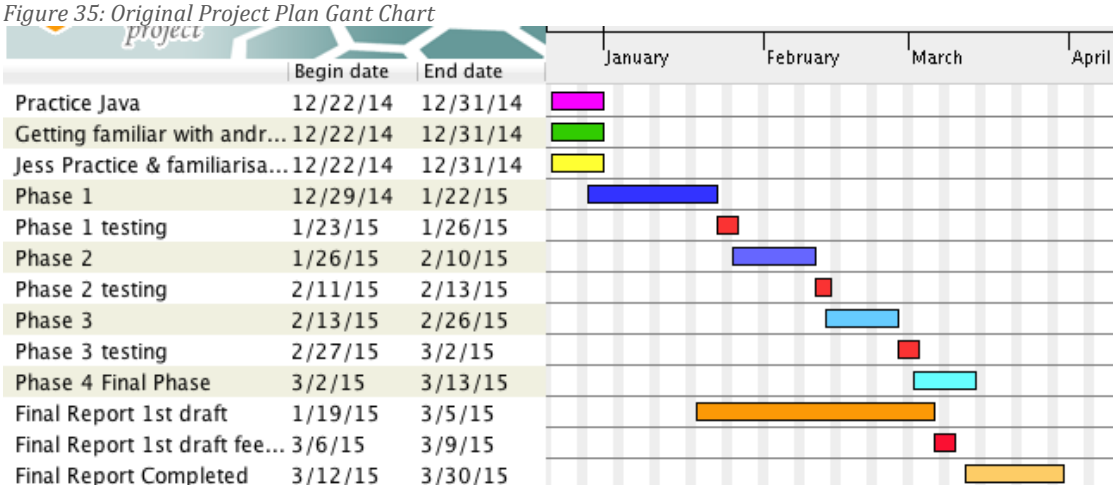
Secondly, Getting the weather information proved to be difficult initially, many tutorials were followed and implemented. However they didn't work efficiently and much of the code involved wasn't understood. The open weather map API was stumbled upon one day while researching, and it proved to be the best solution at the time as it returned the weather information as JSON. The learning outcome here is that quite often the less complex solution is the better one.

Thirdly, using the Google Places API efficiently turned out to be a challenge. Originally all of the activities were going to be displayed on the map at once. However it was later learned that multiple places could not be searched for all at once but just one at a time. To work around this a spinner was added at the top of the screen allowing the user to view and search for each activity one by one. In

the end this approach is actually better as it allows users to view a list of what has been recommended. As a result of this challenge some critical thinking had to be put in place. Learning how to work with loops, arrays and spinners in a way that had previously never been explored was accomplished.

6.2 Plan Analysis

Figure 35: Original Project Plan Gant Chart



The original plan shown in figure 35, proved harder to implement than expected. It was to commence around Christmas time, which was just 2 weeks before the winter examinations resumed (They were split before and after the Christmas period). As a result of this a minimal part of the practice was completed. Phase one commenced a month later than anticipated. Despite the desire to have four phases, there was just one long phase. This final report document itself was to be completed in two phases, however like the development; the plan wasn't followed due to other work commitments in other modules.

Jess rules were to be implemented into the project; however it later proved impossible due to Jess 8 not being released in time. This was unfortunate but this problem was ultimately solved using java, as stated in the interim report.

Chapter 7: Conclusions

7.1 Future Work

There are an abundance of extra features that could be implemented into this project if there were enough time to do so. In this section of the document we will discuss a couple of these ideas that will be implemented into this project at some point in the future.

<u>The Google directions API</u>
This would be a wonderful API to make use of in the application. It would enable the user to get directions to the activities recommended.

Facebook integration
To allow the user to create an account by logging in with Facebook would be an appealing feature to implement, it could also mean allowing the application to take the users information directly from their Facebook such as their age, gender and their likes. With this information activities could be recommended to the user based on their Facebook information.

Displaying extra information
Enabling the user to retrieve more than just the name and vicinity would be great. A place's detail search could be created using the ID's from the place search. Displaying a phone number and description and maybe even a review of an activity could then be possible.

Deploying the application to iOS
It would be great to see this application running on an iPhone, to do this a couple of changes would have to be made to the user interface including adding a back button to the touch screen as Apple devices don't have a physical back button built onto them.

7.2 Overview Of Project And Final Conclusion

The primary aim of this project was to develop an application that would recommend activities to the user based on a number of factors defined earlier in the document. The resulting system has achieved its goal successfully. The application allows the user to create an account, sign in, adjust parameters and be recommended activities based on these parameters along with the time and weather. At the time of writing there were no applications found on the market that recommended activities in the same way.

The focus of this dissertation was to give the reader an insight into how this application came to be. This began with the initial idea and problem to be solved. All of the research that was done for this project was documented and discussed giving the reader an insight as to why certain technologies were chosen over others and why certain decisions were made. The design and architecture chapter explained with the aid of illustrations how the system was designed. The implementation chapter discussed the development and implementation of key features within this application and displayed code snippets were necessary. The validation section explained what was done in order to test the system and the results are shown in the appendix. Extra features that would hopefully be implemented into this project in the future were discussed.

Initially working on this project was tough and felt like an up hill battle. However as the project progressed it became much more enjoyable and felt rewarding. New technologies were learned and programming knowledge increased dramatically. As a result, continuing this kind of work in the future is definitely viewed with more confidence.

Bibliography

[1] "Weotta Go: An iPhone App That Suggests Activities For Right Now | TechCrunch." [Online]. Available: http://techcrunch.com/2012/05/17/weotta-go/. [Accessed: 11-Dec-2014].

[2] "Movie Twist - Android Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=se.saguru.emmovie&hl=en. [Accessed: 11-Dec-2014].

[3] "Weather Based Marketing and Analytics | Skymosity." [Online]. Available: http://www.skymosity.com/. [Accessed: 11-Dec-2014].

[4] "IDC: Smartphone OS Market Share 2014, 2013, 2012, and 2011." [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Accessed: 19-Mar-2015].

[5] A. Allan, *Learning iOS Programming*, 3rd ed. O'Reilly Media, Inc., 2013.

[6] "Swift - Overview - Apple Developer." [Online]. Available: https://developer.apple.com/swift/. [Accessed: 11-Dec-2014].

[7] "Android Phones." [Online]. Available: http://www.android.com/phones/. [Accessed: 23-Mar-2015].

[8] P. Deitel, H. Deitel, and A. Deitiel, *Android How to Program with an Introduction to Java*, 5th ed. Pearson, 2013.

[9] "PhoneGap | About." .

[10] "PhoneGap | Supported Features." [Online]. Available: http://phonegap.com/about/feature/. [Accessed: 11-Dec-2014].

[11] "android - Is PhoneGap a good choice for App Development? - Stack Overflow." .

[12] "Drools - Drools - Business Rules Management System (Java™, Open Source)." [Online]. Available: http://www.drools.org/. [Accessed: 25-Nov-2014].

[13] I. Bratko, *Prolog Programming for artificial intelligence.*, 3rd ed. Pearson, 2001.

[14] "Jess, the Rule Engine for the Java Platform." [Online]. Available: http://herzberg.ca.sandia.gov/. [Accessed: 11-Dec-2014].

[15] "Jess is a rule engine and scripting environment written entirely in Java." [Online]. Available: http://comments.gmane.org/gmane.comp.java.jess/6655. [Accessed: 11-Dec-2014].

[16] "Database Definition." [Online]. Available: http://techterms.com/definition/database. [Accessed: 23-Mar-2015].

[17] "DB-Engines Ranking - popularity ranking of database management systems." [Online]. Available: http://db-engines.com/en/ranking. [Accessed: 23-Mar-2015].

[18] T. M.Connolly and C. E. Begg, *Database Systems A Practical Approach to Design, Implementation, and Management*, 5th ed. Pearson, 2010.

[19] "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean." [Online]. Available:

https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Accessed: 23-Mar-2015].
[20] "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean." [Online]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Accessed: 23-Mar-2015].
[21] "What is JSON: the 3 minute JSON Tutorial (secretGeek.net)." [Online]. Available: http://www.secretgeek.net/json_3mins. [Accessed: 11-Dec-2014].
[22] "Getting Started - Google Maps Android API v2 — Google Developers." [Online]. Available: https://developers.google.com/maps/documentation/android/start#get_an_android_certificate_and_the_google_maps_api_key. [Accessed: 09-Mar-2015].
[23] "Place Search - Google Places API — Google Developers." [Online]. Available: https://developers.google.com/places/documentation/search. [Accessed: 10-Mar-2015].
[24] J. J. A. Denissen, L. Butalid, L. Penke, and M. A. G. van Aken, "The Effects of Weather on Daily Mood: A Multilevel Approach," vol. 8, pp. 662–667, Oct. 2008.
[25] "Connection between Android app and MySQL database using PHP and JSON." [Online]. Available: http://moinur-rahman.blogspot.ie/2012/02/connection-between-android-app-and.html. [Accessed: 30-Mar-2015].
[26] "Showing nearby places using Google Places API and Google Map Android API V2 | Knowledge by Experience." [Online]. Available: http://wptrafficanalyzer.in/blog/showing-nearby-places-using-google-places-api-and-google-map-android-api-v2/. [Accessed: 30-Mar-2015].
[27] "User Interface Design Basics | Usability.gov." [Online]. Available: http://www.usability.gov/what-and-why/user-interface-design.html. [Accessed: 30-Mar-2015].
[28] I. Sommerville, *Software Engineering*, 9th ed. Pearson.
[29] "proto.jpg (470×300)." [Online]. Available: http://grouplab.cpsc.ucalgary.ca/saul/681/1998/prototyping/proto.jpg. [Accessed: 24-Mar-2015].
[30] "jameshom.com | Usability Methods Toolbox | Vertical Prototyping." [Online]. Available: http://usability.jameshom.com/vertical.htm. [Accessed: 24-Mar-2015].
[31] "Layouts | Android Developers." [Online]. Available: http://developer.android.com/guide/topics/ui/declaring-layout.html. [Accessed: 18-Mar-2015].
[32] "Introduction - Google Places API — Google Developers." [Online]. Available: https://developers.google.com/places/documentation/. [Accessed: 09-Mar-2015].
[33] "Getting Started - Google Maps Android API v2 — Google Developers." [Online]. Available: https://developers.google.com/maps/documentation/android/start. [Accessed: 24-Mar-2015].

Appendix

Appendix A: Weather Survey Results

What is your gender?	
Male	66
Female	34

What is your age?					
11- 14	*1*	***25 - 34***	*6*	***55 - 64***	*1*
15 - 17	*3*	***35 - 44***	*1*	***65 - 74***	*1*
18 - 24	*83*	***45 - 54***	*3*	***75+***	*1*

If it were raining outside on your day off, would this put you off going out and doing something?	
Yes	63
No	35

If it were a hot summers day, what would you prefer to do?	
Cinema	2
Walk along the pier	36
Go bowling	1
Climb a mountain	18
Go swimming (Outdoors)	39
Visit a museum	3

If it were a cool winters day, what would you prefer to do?	
Indoor Crazy golf	7
Visit a museum	5

Cinema	62
Go for a long walk	15
Go bowling	7
Aquatic Centre	2

On a scale of 1 - 10 with 10 being the most positive and 1 being the most negative, how does cold weather make you feel?	
Weighted average	4.8

On a scale of 1 - 10 with 10 being the most positive and 1 being the most negative, how does warm weather make you feel?	
Weighted average	8.09

On a scale of 1 - 10 with 10 being the most positive and 1 being the most negative, how does wet and rainy weather make you feel?	
Weighted average	3.26

On a scale of 1 - 10 with 10 being the most positive and 1 being the most negative, how does a heat wave make you feel?	
Weighted average	6.77

Do you think that a mobile app that will recommend activities to you based On your interests, age, gender, location, time of day and the weather is Something you would download?	
Yes	87
No	13

Appendix B: Test Case Results

Test ID	*Description*	*Expected Result*	*Test 1 (March 10)*	*Test 2 (March 23)*
1.	App launches	Launch Screen Opens	PASS	PASS
En	User enters a username	'2' is passed back to the app if the username exists or '1' is passed back if it does not exist.	PASS	PASS
3.	User enters an email address	'3' is passed back to the app if the email exists or '1' is passed back if it does not exist.	PASS	PASS
4.	User confirms their password	The passwords must match in order to proceed with account creation. There is an error if the passwords don't match.	PASS	PASS
5.	User submits their unique account details.	The details entered by the user will be inserted into the MySQL DB.	PASS	PASS
6.	Sign in page launches after account created	User will receive a message stating 'Account successfully created' and be prompted to Sign in.	PASS	PASS
7.	The user signs into the application	'1' is passed back to the app if both username and password match records in the DB. Otherwise '2' is passed back.	PASS	PASS
8.	The app gets the current location.	lat and lon are stored as static Doubles.	PASS	PASS
9.	The weather is displayed	Weather is displayed at the	PASS	PASS

		top of the home screen.		
10.	Clicking on the weather shows more weather	A new page opens displaying even more weather data.	PASS	PASS
11.	The current time is displayed.	The current time is displayed on the home screen with the following format: (HH:MM)	PASS	PASS
12.	Users settings are received into the application.	The users saved settings are displayed on the settings page. If it is their first time to use the app some pre-defined values are set.	FAIL The Application went a bit strange and the settings were random.	PASS
13.	User clicks submit on the 'tweak search settings' page	The DB is updated with the new settings.	PASS	PASS
14.	Age spinner is set to the value of the users age if already defined	The users age range will be set on the spinner the 2nd time they visit the search settings page	FAIL It was always set at 11- 15	PASS
15.	Seekbars are set to value previously set by user	The value of the seek bar(s) is set to the same value of their corresponding variable.	FAIL Set to default.	PASS
16.	Switch is set to previously defined gender.	The value of the switch is set to either male or female.	PASS	PASS
17.	User Signs Out	The user can't press the back button and be brought to the home page once signed out	FAIL Back button brings user back to the home screen.	PASS
18.	The back button is	The user will not accidentally sign	Not Run	PASS

	disabled on the Home Screen	out on the home screen if they keep pressing 'back'		
19.	System checks if the GPS is enables	An if statement can be used to ensure the GPS of the device is enabled before performing a task.	Not Run	PASS
20.	System ensures it is connected to the internet	An if statement can be used to ensure the devide is connected to the internet before performing a task.	Not Run	PASS
22.	Google Camera is looking at users location	When the map launches the users location will be the centre point	Not Run	PASS
23.	Markers are added to the activities location	Markers display at the Lat and Long of the activities	Not Run	PASS

Appendix C: Questionnaire Results

Question	User	Answer	Comment
Did the application launch successfully?	User 1	Yes	
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
Did the "Create Account" button work?	User 1	Yes	
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
Do you think the "Create Account" page is laid out well?	User 1	Yes	
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
Could you create an account successfully?	User 1	Yes	
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
Could you sign in?	User 1	Yes	
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
Sign out of the application. Try to create another account but with the same username. Could you?	User 1	No	
	User 2	No	
	User 3	No	
	User 4	No	
Sign back into the application but intentionally enter the wrong password. Could you sign in?	User 1	No	
	User 2	No	
	User 3	No	
	User 4	No	
Sign in using the correct credentials. Did the weather information display correctly?	User 1	Yes	
	User 2	No	At first it just said 'Null' but it displayed after the settings were submitted.
	User 3	Yes	
	User 4	Yes	
Was the location correct? If not please comment what it said and what it	User 1	No	Maryland, should be churchtown
	User 2	Yes	

should be.	User 3	Yes	
	User 4	No	Rathgar, Should be Rathfarnham
Click "Tweak Search Setting" if it is your first time doing so are the parameters set as follows: female, 18-24, 6km, 50, 50, 50, 50, 50?	User 1	Yes	
	User 2	No	No they are all random.
	User 3	Yes	
	User 4	Yes	
Adjust all of the parameters and click submit. Did it work?	User 1	Yes	
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
Go back into "tweak search settings", are the parameters the same as what you set them to?	User 1	Yes	
	User 2	No	They are the same as before.
	User 3	Yes	
	User 4	Yes	
Go back to the previous page and click the cogwheel at the bottom, attempt to change your password. Could you?	User 1	No	Nothing happened
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
If your password changed, sign out and sign in with your old one. Did it work?	User 1	N/A	
	User 2	No	
	User 3	No	
	User 4	No	
Sign in with your new password. Did it work?	User 1	N/A	
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
Click on "Get Activities" did any activities get recommended? If not tweak the search settings and try again.	User 1	Yes	
	User 2	No	Tweaked settings and then some were recommended.
	User 3	No	3rd time lucky.
	User 4	Yes	
Click on the activity to view a dropdown menu, are there more activities? Please comment the activities that were recommended.	User 1	Yes	Wildlife Park, Science Museum
	User 2	Yes	Trampoline Park, Rock Climbing
	User 3	No	Science Museum
	User 4	Yes	Gym, Ice Skating
If so click on the new activity. Did anything display on the map?	User 1	Yes	
	User 2	Yes	
	User 3	Yes	

	User 4	Yes	
Click on one of the markers. Did it display the name and vicinity?	User 1	Yes	
	User 2	Yes	
	User 3	Yes	
	User 4	Yes	
Do you feel the activities recommended were accurate when compared to the parameters and weather?	User 1	Yes	
	User 2	Yes	It was raining and the recommendations were all indoors
	User 3	Yes	
	User 4	Yes	
Rate the design out of 10	User 1	6/10	Needs a more modern look
	User 2	8/10	
	User 3	7/10	A feature or two missing, like a navigation bar.
	User 4	6/10	Could be improved
Rate the application out of 10	User 1	7.5/10	
	User 2	8/10	
	User 3	7/10	
	User 4	7/10	
Do you feel that there is any features missing that could be added?	User 1	Yes	Information on the activity like a description
	User 2	Yes	Extra information on the activities.
	User 3	Yes	Navigation Bar
	User 4	Yes	Description.

Appendix D: Low Fidelity Prototyping