

# Medical Management Application for Android

**Nicholas Randles**

**May 2016**

**Individual Project**

Submitted in part fulfilment for the degree of

**B.Sc. (Hons) Computing**

School of Informatics and Engineering,  
Institute of Technology Blanchardstown,  
Dublin, Ireland



# Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Degree of **Honours B.Sc. in Computer Science** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed: \_\_\_\_\_

Dated: \_\_\_\_/\_\_\_\_/\_\_\_\_

# Abstract

The purpose of this thesis is to explore the possibility of creating an android application which provides its users with a more efficient way of managing their medical needs than existing application. The project looks at existing applications and sees what they are lacking on and how this project could improve on their mistakes. The project tries to provide a more efficient platform for doctors to communicate with patients and also implement other features which are not very common to give the application an edge over existing applications.

The application developed in this project was developed in Android Studio using Java. The application uses a MySQL database to store user information such as their personal information, medications and appointments. The application allows users to find doctors and add them as contacts or make an appointment. The application allows the user to search for doctors by their name or by location. Finding doctors by location is a very efficient way of finding doctors as the user can find doctors in their area. Unlike most medical applications this application allows patients to book appointment without having to talk to the doctor. The doctors can fill out appointment slots they want to be available and then the user can come along and book one of those appointment slots. This application also provides the users with scheduling tools which allow the users to plan out their week relatively easy, they can enter in their medications and when they want to take them and application will be send them a notification.

The result of this thesis is a fully functioning Android application which provides users with an efficient way to find and communicate with doctors, book medical events, and be reminded of medical events.

# Table of Contents

<b>DECLARATION.....</b>	<b>I</b>
<b>ABSTRACT.....</b>	<b>II</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1: INTRODUCTION .....	1
1.2: AIMS AND OBJECTIVES .....	1
1.3: TECHNOLOGIES REQUIRED .....	2
1.4: BENEFITS .....	2
1.5: MAIN RESEARCH QUESTIONS .....	2
1.6: PROJECT PLAN .....	3
<b>CHAPTER 2: LITERATURE REVIEW.....</b>	<b>1</b>
2.1: INTRODUCTION .....	1
2.2: WHAT IS AN ANDROID APPLICATION? .....	1
2.3: WHY DEVELOP AND ANDROID APPLICATION?.....	2
2.4: WHAT MAKES A GOOD ANDROID APPLICATION?.....	3
2.5: ANDROID APPLICATION FOR MEDICAL MANAGEMENT PURPOSES .....	4
2.6: SECURING USER SENSITIVE DATA IN AN ANDROID APPLICATION .....	4
2.7: CONCLUSION.....	5
<b>CHAPTER 3: METHOD.....</b>	<b>6</b>
3.1: OVERVIEW OF METHOD.....	6
3.1.1: Android Studio IDE .....	6
3.1.2: Android SDK Tools.....	6
3.1.3: Google Play Services SDK .....	6
3.1.4: MySQL.....	6
3.1.5: PHP.....	6
3.1.6: JSON .....	6
3.2: REQUIREMENTS SPECIFICATION AND FEASIBILITY.....	6
3.3: SOFTWARE DEVELOPMENT LIFE CYCLE.....	6
<b>CHAPTER 4: SYSTEM REQUIREMENTS AND SPECIFICATION.....</b>	<b>8</b>
4.1: FUNCTIONAL REQUIREMENTS .....	8
4.1.1: Login Use Case .....	8
4.1.2: Register Use Case .....	9
4.1.3: Select Patient Task Use Case .....	10
4.1.4: Add Medication Use Case .....	11
4.1.5: Search Use Case .....	12

4.1.6: Book Appointment Use Case.....	12
4.1.7: Add Contact Use Case .....	13
4.1.8: Select Doctor Task Use Case .....	14
4.1.9: Add Appointments Use Case .....	14
4.1.10: Check Diary Entry Use Case.....	15
4.1.11: Complete Use Case .....	16
4.2: ACTIVITY DIAGRAM.....	17
4.3: DATA FLOW DIAGRAM.....	18
4.4: SEQUENCE DIAGRAMS.....	19
<b>CHAPTER 5: SYSTEM DESIGN .....</b>	<b>20</b>
5.1: APPLICATION DESIGN.....	20
5.1.2: Scaling .....	20
5.2: USER INTERFACE AND FUNCTIONAL DESIGN.....	20
5.2.1: Welcome Activity .....	20
5.2.2: Login Activity.....	22
5.2.3: Register Activity .....	23
5.2.4: Patient Register Activity.....	24
5.2.5: Doctor Register Activity .....	25
5.2.6: Home Activity.....	26
5.2.7: Search Activity.....	27
5.2.8: Search by Name Activity .....	28
5.2.9: Search by Location Activity .....	29
5.2.10: Calendar Fragment.....	30
5.2.11: Diary Entry Activity.....	31
5.2.12: Contacts Fragment.....	32
5.2.13: Planner Fragment.....	33
5.2.14: Add Medication Activity.....	34
5.2.15: Add Appointments Activity .....	35
5.2.16: Profile Activity .....	36
5.2.17: Available Appointments Activity .....	37
5.3: CLASS DIAGRAM .....	38
5.4: ENTITY RELATIONSHIP DIAGRAM .....	39
<b>CHAPTER 6: IMPLEMENTATION OF PROTOTYPE .....</b>	<b>40</b>
6.1: USER INFORMATION CLASS .....	40
6.2: BACKGROUND TASK CLASS .....	40
6.3: WELCOME ACTIVITY .....	40

6.4: LOGIN ACTIVITY .....	42
6.5: REGISTER .....	43
6.6: REGISTER AS PATIENT .....	44
6.7: REGISTER AS DOCTOR .....	45
6.8: CALENDAR FRAGMENT.....	46
6.9: DIARY ENTRY ACTIVITY .....	47
6.10: PLANNER FRAGMENT.....	48
6.11: ADD MEDICATION ACTIVITY.....	49
6.12: ADD APPOINTMENTS ACTIVITY .....	50
6.13: CONTACTS .....	51
6.14: SEARCH .....	52
6.15: SEARCH BY NAME.....	53
6.16: SEARCH BY LOCATION .....	54
6.17: AVAILABLE APPOINTMENTS .....	55
6.18: PROFILE .....	56
<b>CHAPTER 7: TESTING AND EVALUATION.....</b>	<b>57</b>
7.1: ANDROID API VERSIONS .....	57
7.2: USER TESTING AND RESULTS .....	57
7.2.1: Survey .....	57
7.2.2: Register Test Case .....	58
7.2.3: Login Test Case.....	58
7.2.4: Home Test Case .....	58
7.2.5: Calendar Test Case .....	59
7.2.6: Planner Test Case .....	59
7.2.7: Contacts Test Case .....	59
7.3: EVALUATION.....	59
<b>CHAPTER 8: CONCLUSIONS AND FURTHER WORK.....</b>	<b>60</b>
8.1: CONCLUSION.....	60
8.2: FURTHER WORK .....	60
<b>APPENDIX A: PROJECT PLANNING .....</b>	<b>61</b>
<b>LIST OF REFERENCES.....</b>	<b>62</b>

# List of Figures

Figure 1.1: Gantt chart .....	3
Figure 2.1: Android Platform Architecture [5] .....	1
Figure 2.2: Global growth of mobile devices [6] .....	2
Figure 2.3: Operating System market shares [7].....	3
Figure 3.1: Spiral Model [29].....	7
Figure 4.1: Login - Use Case .....	8
Figure 4.2: Register – Use Case .....	9
Figure 4.3: Select Patient Task Use Case .....	10
Figure 4.4: Add Medication Use Case .....	11
Figure 4.5: Search Use Case .....	12
Figure 4.6: Book Appointment Use Case.....	12
Figure 4.7: Add Contact Use Case .....	13
Figure 4.8: Select Doctor Task Use Case .....	14
Figure 4.9: Add Appointments Use Case .....	14
Figure 4.10: Check Diary Use Case .....	15
Figure 4.11: Complete Use Case .....	16
Figure 4.12: Activity Diagram .....	17
Figure 4.13: Data Flow Diagram.....	18
Figure 4.14: Login Sequence Diagram .....	19
Figure 4.15: Register Sequence Diagram .....	19
Figure 5.1: Welcome activity design .....	21
Figure 5.2: Login activity design .....	22
Figure 5.3: Register activity design .....	23
Figure 5.4: Patient Register activity design .....	24
Figure 5.5: Doctor Register activity design.....	25
Figure 5.6: Home activity design.....	26
Figure 5.7: Search activity design.....	27
Figure 5.8: Name Search activity design .....	28
Figure 5.9: Location Search activity design .....	29
Figure 5.10: Calendar fragment design .....	30
Figure 5.11: Diary Entry activity design .....	31
Figure 5.12: Contacts fragment design .....	32
Figure 5.13: Planner fragment design .....	33
Figure 5.14: Add Medication activity design .....	34
Figure 5.15: Add Appointments activity design .....	35
Figure 5.16: Profile activity design .....	36
Figure 5.17: Available Appointments activity design .....	37
Figure 5.18: Class Diagram .....	38
Figure 5.19: Entity Relationship Diagram .....	39
Figure 6.1: Welcome activity implementation .....	41
Figure 6.2: Login activity implementation.....	42
Figure 6.3: Register activity implementation .....	43
Figure 6.4: Patient Register activity implementation.....	44
Figure 6.5: Doctor Patient activity implementation .....	45
Figure 6.6: Calendar fragment implementation.....	46

Figure 6.7: Diary Entry activity implementation .....	47
Figure 6.8: Planner fragment implementation.....	48
Figure 6.9: Add Medication activity implementation.....	49
Figure 6.10: Add Appointments activity implementation .....	50
Figure 6.11: Contacts fragment implementation .....	51
Figure 6.12: Search activity implementation .....	52
Figure 6.13: Name Search activity implementation .....	53
Figure 6.14: Location Search activity implementation.....	54
Figure 6.15: Profile activity implementation.....	56
Figure 6.16: Available Appointments activity implementation .....	55



# List of Tables

Table 1.1: Work Breakdown Structure .....	3
Table 4.1: Login – Use Case Specification .....	8
Table 4.2: Register – Use Case Specification.....	9
Table 4.3: Select Patient Task Use Case Specification .....	10
Table 4.4: Add Medication Use Case Specification .....	11
Table 4.5: Search Use Case Specification.....	12
Table 4.6: Book Appointment Use Case Specification.....	13
Table 4.7: Add Contact Use Case Specification .....	13
Table 4.8: Select Doctor Task Use Case Specification .....	14
Table 4.9: Add Appointments Use Case Specification.....	15
Table 4.10: Check Diary Use Case Specification.....	15
Table 7.1: Survey.....	57
Table 7.2: Register Test Case .....	58
Table 7.3: Login Test Case .....	58
Table 7.4: Home Test Case .....	58
Table 7.5: Calendar Test Case .....	59
Table 7.6: Planner Test Case .....	59
Table 7.7: Contacts Test Case.....	59

# Chapter 1: Introduction

## 1.1: Introduction

Technology has really changed the world we live in. It has helped make our lives easier in so many ways. It has made it much easier for us to be more organised and communicate with each other. There are many growing trends in technology these days that have helped us to be more organised and communicate with each other. No one trend has helped us accomplish this more than mobile applications. Mobile applications have provided us with all different types of services such as calendars, alarms and instant messaging.

These services have become very helpful for people in many different scenarios but have not been taken advantage of in others. These days many medical practices do not use technology as well as they could to communicate with their patients. Many medical practises have not been very efficient with the way they assign appointments to patients. Several patients are often given the same appointment times. This can cause appointment rooms to be overcrowded. This can be a health risk to vulnerable patients. It also can be a big inconvenience for others who may be waiting long periods of time. Also many patients do not use technology as well as they could to help them with their illnesses. Many patients could benefit with using an application to remind them when to take their medicine.

There have been many attempts to develop an application which helps people manage their medical needs. Most of which provide a good service but have their shortcoming. One of the applications I discovered was “Book Dr Appointment” [1]. This application allows new users to join and find a Doctor. They can then book a doctor appointment with the Doctor of their chose. This application has been quite successful amongst android users. It's has a 4.5/5 on the Google play store. It has been installed by 5,000 – 10,000 android users.

There are many other applications on the Google play store just like “Book Dr Appointment” that all pretty much do the same thing. Most of them do not implement any additional features which could be beneficial to the doctor or to the patient. Instead of just recreating the wheel, this project will explore the possibility of developing an android application to help users manage their medical needs in a different and more efficient way than the ways that are already available. Some of these additional features will include a scheduling system which will allow doctors and patients to manage their schedules in an efficient way.

## 1.2: Aims and Objectives

The aims and objectives for this project are as follows:

- Research mobiles applications in relation to medicine and learn the best approach to take.
- Develop a complete operational Android application that allows doctors and patients to communicate, manage appointments and medications in an efficient way. The application should implement the following features:
  - Allow patients to book appointments.
  - Allow patients to communicate with their doctor through phone.
  - Allow patients to fill out their medication schedule.

- Remind patients when they have an upcoming appointment.

### **1.3: Technologies Required**

The technologies required to develop the application are as follows:

- Android Studio IDE – Will be used as the development environment for this project.
- Android SDK Tools – Essential software tools for the applications development in android.
- Google Play Services – Application will use important google service such as maps.
- MySQL - MySQL database will be used to store the important data such as appointment times.
- PHP – PHP scripts will be used to communicate with the database.
- JSON – Database information will be transferred in JSON format.
- Android Device – Device will be used to run and test the application.

### **1.4: Benefits**

This application will be very helpful for everyone who uses it. Some of the ways it will help include:

- It will help doctors and their patients to have better relationships by helping them to communicate with each other better.
- It will provide patients with a more efficient way of booking appointments with their doctors.
- It will help increase patients chances of them being seen at their scheduled time by preventing problems such as double bookings.
- It will help prevent patients from missing their appointments by sending them a reminder.
- It will help prevent overcrowding in medical practices as doctors and patients will be more likely to meet at their scheduled time.
- It will also help remind patients to take their medications by sending them a notifications.

### **1.5: Main Research Questions**

- Determine if this application is unique.
- Determine is this application improves on existing applications in the medical field.
- Determine if there is an actual need for an application like this in medical practices.

## 1.6: Project Plan

The project has been broken down into separate manageable tasks that will be conducted from now till the deadline. Each task is named and described in table 1 and they are mapped out on a Gantt chart (figure 2) according to their dates. The purpose of this work breakdown structure is to provide a guideline for schedule development and control.

1	Project Proposal	A document that contains information about the project and task that need to be completed.
2	Research on similar application	Research on existing applications that involve scheduling appointments.
3	Literature Review	An evaluative report on the information found in the literature relating to patient scheduling.
4	Risk analysis	Analyse the risk involved in taking on this project.
5	Cost assessment	Analyse the potential cost of the project.
6	Application design	Start designing the application. Conduct wire framing.
7	Develop application	Start developing the application with the necessary resources.
8	Application testing	Test the application to discover any existing problems.
9	Document results	Document findings during testing process.
10	Final application	Fix any problems in application and get working 100%.
11	Documentation	Put all documents together. Finish off thesis.
12	Prepare presentation	Prepare for presentation through notes and slides.

Table 1.1: Work Breakdown Structure

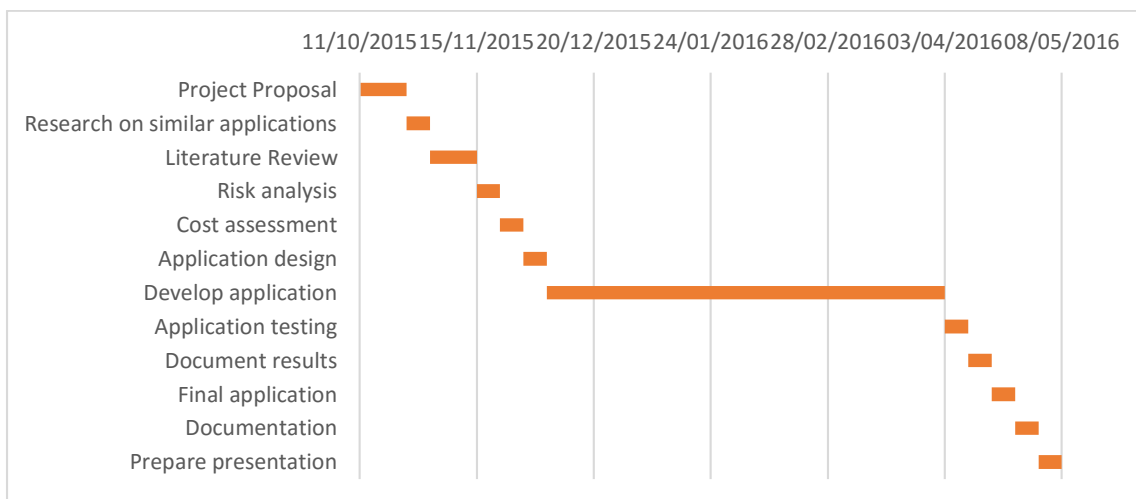


Figure 1.1: Gantt chart

# Chapter 2: Literature Review

## 2.1: Introduction

The focus of this literature review is to explore the possibility of creating an Android application which can help users' to manage their medical needs in an efficient way. This literature review will focus in on five main topics of research. The five topics are as follows:

- What is an Android application?
- Why develop and Android application?
- What makes a good Android application?
- Android Application for medical management purposes
- Securing user sensitive data in an Android application

## 2.2: What is an Android application?

Android is an open source operating system that is based on the Linux platform. It is composed of an operating system, middleware, user interface and application software. It is mainly used for mobile devices and tablets. It is developed by the Open Handset Alliance which is composed of over 30 technology companies, some of which include Google, HTC, Motorola, Samsung and LG [2]. Android promotes the Google enterprise target achievement which is "provide information for everyone at any time in any place" [3]. According to [4], Android has the following 5 characteristics:

- It is a completely open and free mobile phone software platform available to all developers.
- The android platform has no boundaries. Applications are able to access the core mobile device with the standard API.
- All applications on Android are equal. This means they can be replaced and extended.
- Applications can be embedded into JavaScript and HTML.
- Applications are able to run at the same time since Android is a complete multi-task environment.

The Android platform architecture can be split into four layers [5]. They are illustrated in Figure 1 from highest to lowest.

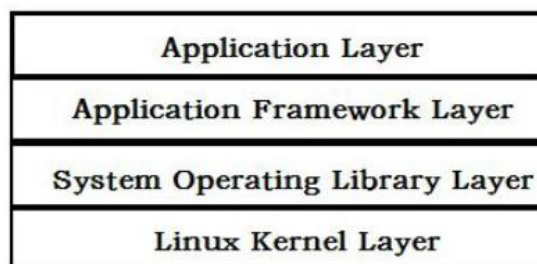


Figure 2.1: Android Platform Architecture [5]

1. Application layer: This layer refers to the programs that are written in Java and run in the virtual machine. For example, Google provides many applications into the system such as a calendar and a browser.
2. Application Framework layer: This layer refers to the API frameworks which are provided by Google developers to help other developers to create their own applications easier through these frameworks.
3. System Operating Library Layer: This layer refers to the libraries provided to developers to help them with the application development. Examples of this include OpenGL and SQLite.
4. Linux Kernel Layer: This layer refers to the Linux kernel, which is the layer between the hardware and software stacks. It provides core services such as memory and process management, security and driving model.

### 2.3: Why develop and Android application?

Mobile application development is the fastest growing trend in the Information Technology industry. There is a high demand for mobile application development and this is supported by [6] who have shown that mobile ownership overtook desktop ownership in 2014. This is illustrated in Figure 2.

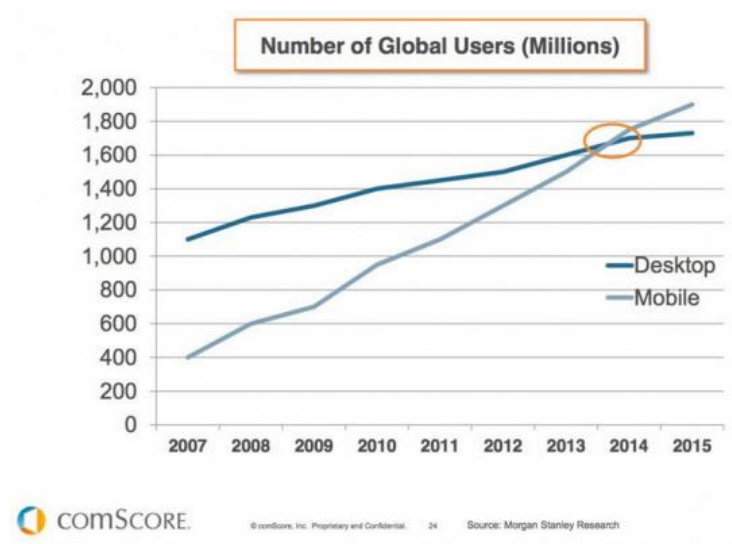


Figure 2.2: Global growth of mobile devices [6]

There is a wide range of different mobile operating system to choose from. In order to choose which operating system is the best chose we need to look at trends and advantages some may have over others. According to [7] Android had the majority of the market share at 52% in the United States in 2013. This is illustrated in Figure 3.

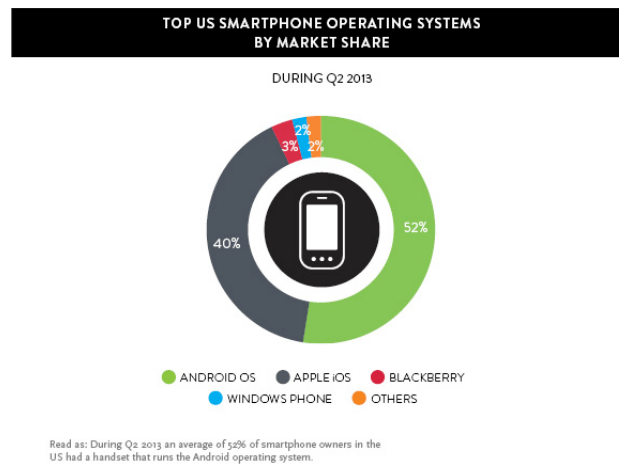


Figure 2.3: Operating System market shares [7]

Not only is the user base much bigger than other operating systems but the developer tools are better according to [8]. [8] said that Apple app developers have started to cross over to Android because it provides better tools for its developers and is also a lot more flexible than Apple OS development. Apple have a very strict policy when it comes to developers trying to get their applications onto iOS. Unlike Apple, Android is open and its software developer kit helps developers get their applications across as many platform as possible [9].

## 2.4: What makes a good Android application?

Privacy is a very important part of mobile applications for mobile device users. Many mobile device users believe it is especially important when it comes to dealing with personal health related information. It is important that mobile application developers develop privacy policies in meaningful ways that meet the requirements of their intended target audience [10]. The problem with most privacy policies is that they are written in a way which makes it hard for the user to understand or makes them completely ignore them. A survey of 584 university students showed that most of them do not read privacy policies completely, but they would be worried if their information was shared with a third party [11].

Users have lost trust in applications over the years as stories come out about personal data being shared secretly and illegally with advertisers [12]. The U.S Federal Trade Commission (FTC) announced that after conducting a basic investigation into 12 popular health and fitness applications, they found that they were sharing their users' personal data with 76 different third parties [13].

Developers now have to gain their users' trust by clearly explaining the data their collecting from them, how it's being used and who it's being shared with. They need to take into account their users' reading comprehension levels when they create their privacy policies. Failure to understand the privacy policy could lead to the user sharing personal health information that they did not intend to, or it could prevent the user from installing the application due to their privacy concerns. Readability tests can be conducted to examine the difficulty of the vocabulary and the structure of the sentences in the privacy policy. It is recommended that documents for health instructions be written at a 5<sup>th</sup> grade level [14].

In Android, applications must request to share resources, data and device features such as GPS, camera and contacts. [15] says that there is evidence to support that some users are

sometimes confused by this application permission model. Users can review the necessary permissions required for an application in the Google Play Store before they download it [16].

## **2.5: Android Application for medical management purposes**

Mobile medical applications have really changed the health care system for the better over the last few years. These advancements are really needed in this day and age because of the increasing aging population of the world. They are providing doctors and their patients a number tools and resources to help them manage their health [17]. A survey at a Canadian medical school found that 85% of students owned smartphones and often used medical apps to reference disease diagnoses and medications [18]. A review on academic literature by [19] found most apps targeted at patients focused on certain health conditions, health diaries and medical calculators.

There are many different approaches to medical applications out there, one research paper [20], discussed an application to monitor a patients vitals. [20] allows doctors to view up to date information of the patients' vital health parameters such as, heart rate and blood pressure. The doctors can monitor these results and make notes on them. If the patients vital levels go below a critical limit their doctor will be notified that their patient is in danger.

Another medical application approach was discussed in the research paper [21]. [21] is a mobile application which aims to stop the common problem of misinterpreted drug prescriptions. This has been a serious problem in the medical field which has led to the deaths of thousands of people [22]. It can be caused by the doctor's sloppy handwriting or inability of the pharmacist to correctly interpret the drug name on the prescription. This application allows users to take a picture of the prescription with their phone camera. The picture is then processed by the handwriting recognition algorithm and returns the matching result from the database.

## **2.6: Securing user sensitive data in an Android application**

Since Android is the leading platform of the smartphone market it is a big target by many computer criminals [23]. Smartphones often contain important information such as medical information and it is important that this information is not leaked as it can be a serious loss to the android smartphone user. To prevent private information being leaked Android uses a different technique compared to more typical operating systems that rely on third party applications. Android have their own built in security scheme instead [24].

Not all users stay up to date with the most current version of Android though, this means they are more vulnerable as they have not received the necessary security updates. [25] says that only 13.6% of all android devices are using the latest version of android.

Security is divided into three categories, they are as follows:

- Vulnerability: This is a weakness in the system that may be a security threat [26].
- Threat: This is everything that can be done to harm the system by using the vulnerability [27].
- Control: This is the process of trying to stop the harm of the threat [27].



If the security is perfect there will be no serious threats or vulnerabilities to take advantage of.

Some of the security features of Android which help developers to build secure applications include the following [28]:

- The Android Application Sandbox: This isolates your application's data and code execution from other applications.
- An application framework that has strong security functionality including permissions, secure IPC and cryptography.
- An encrypted filesystem which the user can be enabled when their device is lost or stolen to protect important data.

## **2.7: Conclusion**

Based upon the research that has been currently undertaken in the field of Android applications it seems like it is very possible to create an Android application which enables users to efficiently manage their medical needs. The information in this literature review will have a very positive impact on the development of our Android application. This literature review has shown us a detailed look on how android applications are developed and why they should be developed over other mobile application options. It has shown us what it takes to create a good application by implementing user friendly aspects such as a privacy policy. We have seen security issues in Android and ways to resolve them. This will help us to make our application more secure. This literature review has also provided us with different approaches that other developers took to create medical applications in Android. We can take these approaches and learn from them and try to improve on their mistakes.

# Chapter 3: Method

## 3.1: Overview of Method

### 3.1.1: Android Studio IDE

The Android studio IDE was downloaded from the android website [ref]. It is the official integrated development environment for android. It was chosen over eclipse because of its gradle build system, improved visual editor and better code completion. The application was built completely in Android Studio using the Android software development tools.

### 3.1.2: Android SDK Tools

The Android SDK was downloaded to develop the application. The Android SDK provides many different packages such as the SDK Tools. This package provides tools for debugging and testing, and other utilities that are needed to develop applications. The application was build, tested and debugged using the Android SDK tools.

### 3.1.3: Google Play Services SDK

The Google play services was downloaded so that the application could take advantage of the latest Google powered APIs such as Google Maps. The application used Google Maps to allow patients to search for doctors by location using a map.

### 3.1.4: MySQL

MySQL is a relational database management system. The application used a MySQL database to store data such as the doctors and patients personal information, appointments time and medications.

### 3.1.5: PHP

PHP is a server-side scripting language. The application used PHP scripts to communicate with the database. Through PHP scripts the application is able to request data from the database and can also send data to the database.

### 3.1.6: JSON

JSON is a lightweight, human readable format for structuring data. JSON was chosen over XML mainly because of its ability to generally parse data faster than XML. The application used JSON to read in data from the database.

## 3.2: Requirements Specification and Feasibility

The application was developed with Android 4.0 (Ice Cream Sandwich) which means it will run on 94% of all android devices. The application has a minimum SDK version of 14 and a target SDK version of 19. The application was tested on the following devices:

- Sony Xperia E – Running on Android 4.0 (Ice Cream Sandwich)
- Samsung S5 – Running on Android 5.0 (Lollipop)
- Nexus 10 – Running on Android 6.0 (Marshmallow)

## 3.3: Software Development Life Cycle

The Software Development Life Cycle which was chosen for the project is the Spiral model. This model was chosen as it involves a lot of testing which will help the final application to be

better. The spiral model is similar to the waterfall model but it is an upgraded version. The spiral model consists of four phases. They are Planning, Risk Analysis, Engineering and Evaluation. The project goes through these phases in iteration. This approach assumes that no one gets it right the first time, each iteration refines the previous result.

- Planning Phase: In this phase requirements and objectives are gathered.
- Risk Analysis Phase: In this phase risks are identified and are analysed. Alternative solutions are also evaluated.
- Engineering Phase: In this phase the software is developed and testing is carried out.
- Evaluation Phase: In this phase the project output is evaluated before its takes its next spiral.

The main advantage of the spiral model is that it reduces the chances of project failure. This is achieved through extensive risk analysis. By carrying out risk analysis it can help avoid risks which could cause the project to fail.

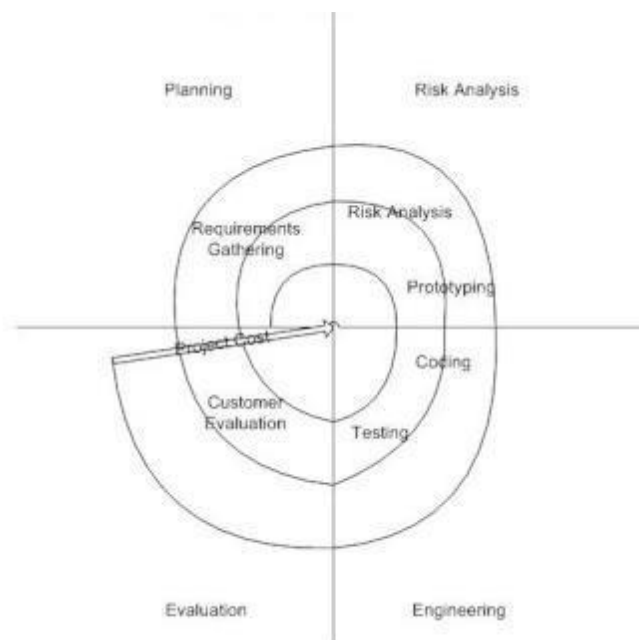


Figure 3.1: Spiral Model [29]

# Chapter 4: System Requirements and Specification

## 4.1: Functional Requirements

### 4.1.1: Login Use Case

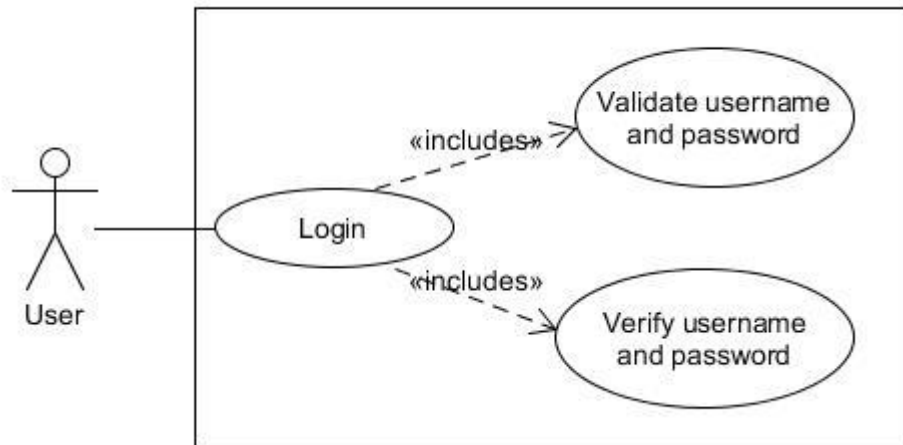


Figure 4.1: Login - Use Case

Use Case Name	Login
Description	This use case describes how a user logs in to the application.
Primary Actor	User
Main Flow	<p>This use case starts when a user is not logged in and chooses to login.</p> <ol style="list-style-type: none"><li>1. The user enters their username and password.</li><li>2. The user submits their username and password.</li><li>3. The application validates the username and password.</li><li>4. The application verifies the username and password.</li><li>5. The application displays the user's home screen.</li></ol>
Alternative Flow	<p>3a. Missing username or password.</p> <ol style="list-style-type: none"><li>1. The application displays "missing username or password" message.</li><li>2. Use case resumes at main flow step 1.</li></ol> <p>4a. Invalid username or password.</p> <ol style="list-style-type: none"><li>1. The application displays "invalid username or password" message.</li><li>2. Use case resumes at main flow step 1.</li></ol>
Preconditions	<ol style="list-style-type: none"><li>1. The application is launched.</li><li>2. User has valid account.</li><li>3. Network connection is available.</li></ol>
Postconditions	<ol style="list-style-type: none"><li>1. The application displays the user's home screen.</li></ol>

Table 4.1: Login – Use Case Specification

#### 4.1.2: Register Use Case

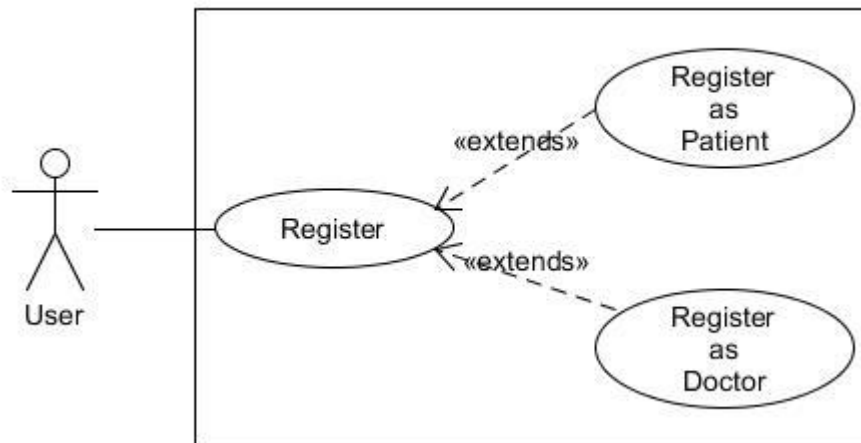


Figure 4.2: Register – Use Case

Use Case Name	Register
Description	This use case describes how a user registers in the application.
Primary Actor	User
Main Flow	<p>This use case starts when a user is not logged in and chooses to register.</p> <ol style="list-style-type: none"><li>1. The user chooses to register as a patient.</li><li>2. The user enters in their details.</li><li>3. The user submits their information.</li><li>4. The application validates the user's information.</li><li>5. The application verifies the user's information.</li><li>6. The application displays the user's home screen.</li></ol>
Alternative Flow	<p>1a. The user chooses to register as a doctor.</p> <p>3a. Missing information.</p> <ol style="list-style-type: none"><li>1. The application displays "missing information" message.</li><li>2. Use case resumes at main flow step 2.</li></ol> <p>3b. Incorrect input format</p> <ol style="list-style-type: none"><li>1. The application displays "incorrect input format" message.</li><li>2. Use case resumes at main flow step 2.</li></ol> <p>4a. Username already taken.</p> <ol style="list-style-type: none"><li>1. The application displays "username taken" message.</li><li>2. Use case resumes at main flow step 2.</li></ol>
Preconditions	<ol style="list-style-type: none"><li>1. Register screen is displayed.</li><li>2. User is not signed in.</li><li>3. Network connection is available.</li></ol>
Postconditions	<ol style="list-style-type: none"><li>1. The application displays the user's home screen.</li></ol>

Table 4.2: Register – Use Case Specification

### 4.1.3: Select Patient Task Use Case

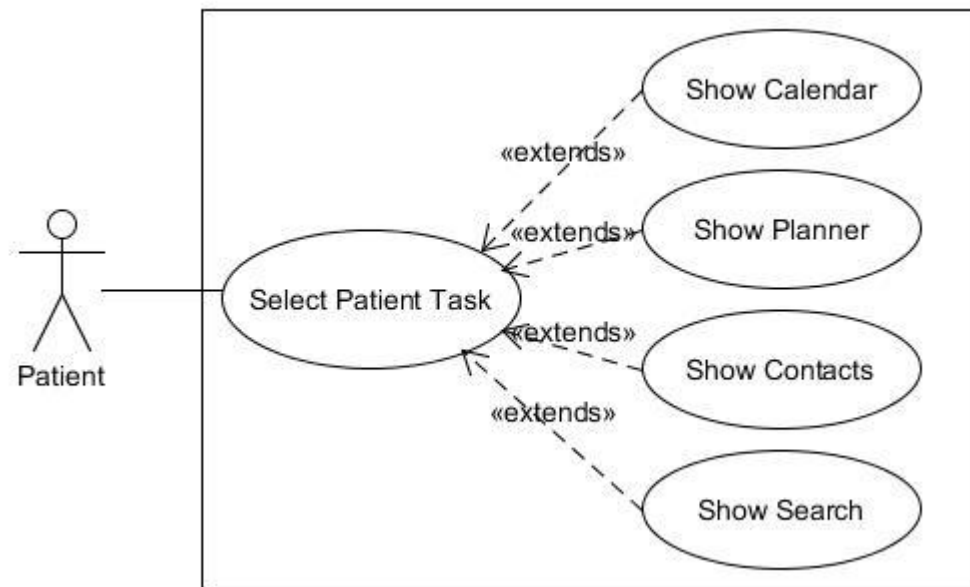


Figure 4.3: Select Patient Task Use Case

Use Case Name	Select Patient Task
Description	This use case describes how a patient selects tasks in the application.
Primary Actor	Patient
Main Flow	<p>This use case starts when a patient is logged in and on the home screen.</p> <ol style="list-style-type: none"> <li>1. The application displays options to the patient.</li> <li>2. The patient selects the calendar.</li> <li>3. The application displays the patient's calendar screen.</li> </ol>
Alternative Flow	<ol style="list-style-type: none"> <li>2a. The patient selects the planner.               <ol style="list-style-type: none"> <li>1. The application displays the patient's planner screen.</li> </ol> </li> <li>2b. The patient selects the contacts.               <ol style="list-style-type: none"> <li>1. The application displays the patient's contacts screen.</li> </ol> </li> <li>2c. The patient selects the search.               <ol style="list-style-type: none"> <li>1. The application displays the patient's search screen.</li> </ol> </li> <li>2d. The patient selects log out.               <ol style="list-style-type: none"> <li>1. The application logs the patient out.</li> </ol> </li> </ol>
Preconditions	<ol style="list-style-type: none"> <li>1. Logged in as patient.</li> <li>2. Patient is on home screen.</li> </ol>
Postconditions	<ol style="list-style-type: none"> <li>1. The application display the screen selected by the patient.</li> </ol>

Table 4.3: Select Patient Task Use Case Specification

#### 4.1.4: Add Medication Use Case

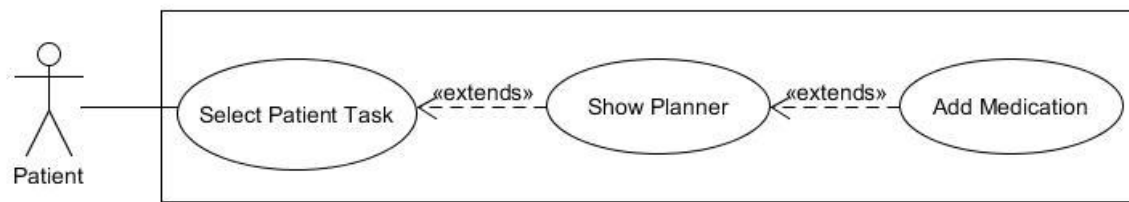


Figure 4.4: Add Medication Use Case

Use Case Name	Add Medication
Description	This use case describes how a patient adds medication in the application.
Primary Actor	Patient
Main Flow	<p>The use case starts when a patient is logged in and selects add medication on the planner screen.</p> <ol style="list-style-type: none"> <li>1. The application displays an add medication form.</li> <li>2. The patient enters the name, start date, time and duration for the medication.</li> <li>3. The patient submits the information.</li> <li>4. The application validates the information.</li> <li>5. The application saves the information.</li> </ol>
Alternative Flow	<p>4a. Missing information.</p> <ol style="list-style-type: none"> <li>1. The application displays “missing information” message.</li> <li>2. Use case resumes at main flow step 2.</li> </ol> <p>4b. Incorrect input format</p> <ol style="list-style-type: none"> <li>1. The application displays “incorrect input format” message.</li> <li>2. Use case resumes at main flow step 2.</li> </ol> <p>5a. The application is unable to save information</p> <ol style="list-style-type: none"> <li>1. The application displays “try again” message.</li> <li>2. Use case resumes at main flow step 3.</li> </ol>
Preconditions	<ol style="list-style-type: none"> <li>1. Logged in as patient.</li> <li>2. Patient is on planner screen.</li> <li>3. Network connection is available</li> </ol>
Postconditions	<ol style="list-style-type: none"> <li>1. Patient’s medication information is added to application.</li> </ol>

Table 4.4: Add Medication Use Case Specification

#### 4.1.5: Search Use Case

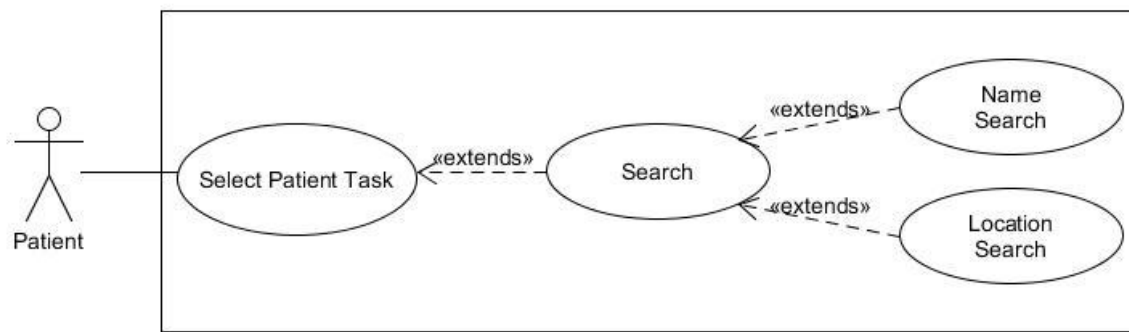


Figure 4.5: Search Use Case

Use Case Name	Select
Description	This use case describes how a patient searches for doctors in the application.
Primary Actor	Patient
Main Flow	<p>This use case starts when patient selects the search feature on the home screen.</p> <ol style="list-style-type: none"> <li>1. The patient chooses to search by name.</li> <li>2. The patient enters the doctor's name.</li> <li>3. The application returns doctors with similar names.</li> <li>4. The patient selects a doctor.</li> <li>5. The application displays the doctor's profile</li> </ol>
Alternative Flow	<ol style="list-style-type: none"> <li>1a. The user chooses to search by location.               <ol style="list-style-type: none"> <li>1. The patient enters in their address and max distance.</li> <li>2. The application displays doctors within the max distance of the address.</li> <li>3. Use case resumes at main flow step 3.</li> </ol> </li> <li>3a. The application returns no doctors.               <ol style="list-style-type: none"> <li>1. The application displays "no doctors found" message.</li> <li>2. Use case resumes at main flow step 2.</li> </ol> </li> </ol>
Preconditions	<ol style="list-style-type: none"> <li>1. Logged in as patient.</li> <li>2. Patient is on planner screen.</li> <li>3. Network connection is available</li> </ol>
Postconditions	<ol style="list-style-type: none"> <li>1. The patient finds a doctor</li> </ol>

Table 4.5: Search Use Case Specification

#### 4.1.6: Book Appointment Use Case

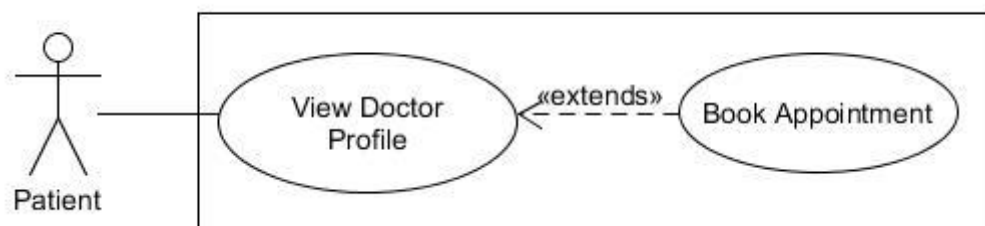


Figure 4.6: Book Appointment Use Case



Use Case Name	Book Appointment
Description	This use case describes how a patient book appointments with doctors in the application.
Primary Actor	Patient
Main Flow	<p>This use case starts when a patient is logged in and is on a doctor's page and chooses to book an appointment.</p> <ol style="list-style-type: none"> <li>1. The patient selects a date.</li> <li>2. The application returns available appointments scheduled that day.</li> <li>3. The user selects an appointment.</li> <li>4. The application saves that appointment for the patient.</li> </ol>
Alternative Flow	<p>3a. The application returns no appointments.</p> <ol style="list-style-type: none"> <li>1. The application displays "no appointments available" message.</li> <li>2. Use case resumes at main flow step 1.</li> </ol>
Preconditions	<ol style="list-style-type: none"> <li>1. Logged in as patient.</li> <li>2. Patient is on planner screen.</li> <li>3. Network connection is available</li> </ol>
Postconditions	<ol style="list-style-type: none"> <li>1. The patient books an appointment.</li> </ol>

Table 4.6: Book Appointment Use Case Specification

#### 4.1.7: Add Contact Use Case

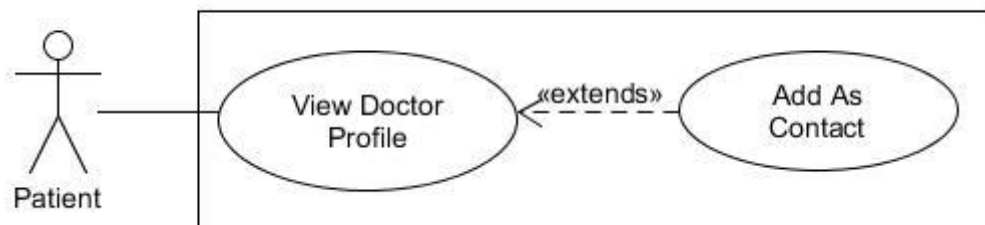


Figure 4.7: Add Contact Use Case

Use Case Name	Add Contact
Description	This use case describes how a patient adds a doctor as a contact in the application.
Primary Actor	Patient
Main Flow	<ol style="list-style-type: none"> <li>1. This use case starts when a patient is logged in and is on a doctor's page.</li> <li>2. The patient selects add a contact.</li> <li>3. The application saves the contact relationship.</li> </ol>
Alternative Flow	<p>3a. Contact already added.</p> <ol style="list-style-type: none"> <li>1. The application displays "contact already added" message.</li> </ol>
Preconditions	<ol style="list-style-type: none"> <li>1. Logged in as patient.</li> <li>2. Patient is on planner screen.</li> <li>3. Network connection is available</li> </ol>
Postconditions	<ol style="list-style-type: none"> <li>1. The patient adds doctor to contacts.</li> </ol>

Table 4.7: Add Contact Use Case Specification

#### 4.1.8: Select Doctor Task Use Case

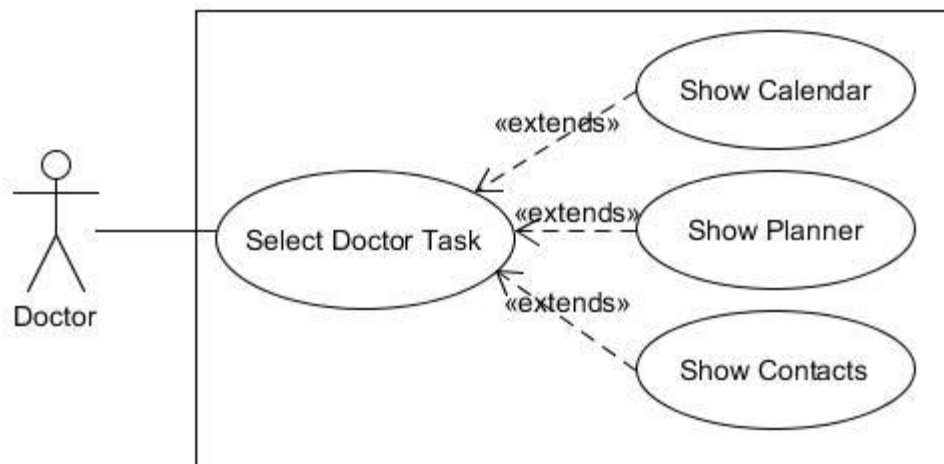


Figure 4.8: Select Doctor Task Use Case

Use Case Name	Select Doctor Task
Description	This use case describes how a doctor selects tasks in the application.
Primary Actor	Doctor
Main Flow	<ol style="list-style-type: none"> <li>1. The application displays options to the doctor.</li> <li>2. The doctor selects the calendar.</li> <li>3. The application displays the doctor's calendar screen.</li> </ol>
Alternative Flow	<ol style="list-style-type: none"> <li>2a. The doctor selects the planner.               <ol style="list-style-type: none"> <li>1. The application displays the doctor's planner screen.</li> </ol> </li> <li>2b. The doctor selects the contacts.               <ol style="list-style-type: none"> <li>1. The application displays the doctor's contacts screen.</li> </ol> </li> <li>2c. The doctor selects log out.               <ol style="list-style-type: none"> <li>1. The application logs the doctor out.</li> </ol> </li> </ol>
Preconditions	<ol style="list-style-type: none"> <li>1. Logged in as doctor.</li> <li>2. Doctor is on home screen.</li> </ol>
Postconditions	<ol style="list-style-type: none"> <li>1. The application displays the screen selected by the doctor.</li> </ol>

Table 4.8: Select Doctor Task Use Case Specification

#### 4.1.9: Add Appointments Use Case

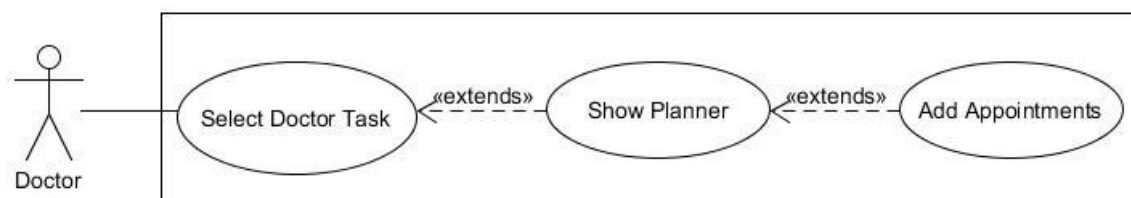


Figure 4.9: Add Appointments Use Case

Use Case Name	Add Appointments
Description	This use case describes how a doctors adds appointments in the application.
Primary Actor	Doctor
Main Flow	<p>The use case starts when a doctor is logged in and selects add appointments on the planner screen.</p> <ol style="list-style-type: none"> <li>1. The application displays an add appointments form.</li> <li>2. The doctor enters the schedule name, start date, start time, duration of each appointment and the duration of the schedule.</li> <li>3. The doctor submits the information.</li> <li>4. The application validates the information.</li> <li>5. The application saves the information.</li> </ol>
Alternative Flow	<p>4a. Missing information.</p> <ol style="list-style-type: none"> <li>3. The application displays “missing information” message.</li> <li>4. Use case resumes at main flow step 2.</li> </ol> <p>4b. Incorrect input format</p> <ol style="list-style-type: none"> <li>3. The application displays “incorrect input format” message.</li> <li>4. Use case resumes at main flow step 2.</li> </ol> <p>5a. The application is unable to save information</p> <ol style="list-style-type: none"> <li>3. The application displays “try again” message.</li> <li>4. Use case resumes at main flow step 3.</li> </ol>
Preconditions	<ol style="list-style-type: none"> <li>1. Logged in as doctor.</li> <li>2. Doctor is on planner screen.</li> <li>3. Network connection is available</li> </ol>
Postconditions	<ol style="list-style-type: none"> <li>1. Doctor’s appointment schedule is added to the application.</li> </ol>

Table 4.9: Add Appointments Use Case Specification

#### 4.1.10: Check Diary Entry Use Case

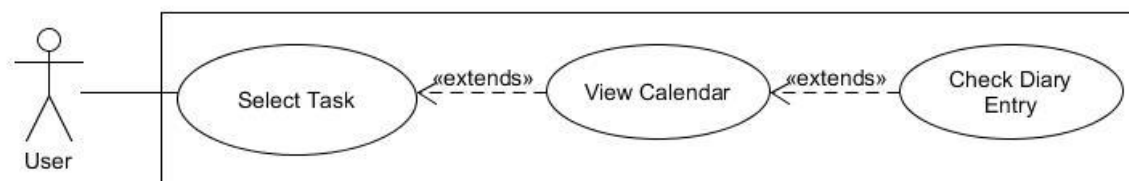


Figure 4.10: Check Diary Use Case

Use Case Name	Check Diary Entry
Description	This use case describes how a user check diary entries in the application.
Primary Actor	User
Main Flow	<p>This use case starts when a user is logged in and selects a date on the calendar screen.</p> <ol style="list-style-type: none"> <li>1. The application displays appointments for date selected.</li> <li>2. The user selects an event.</li> <li>3. The application displays more information about event.</li> </ol>
Alternative Flow	<p>1a. The application displays medications for date selected.</p>
Preconditions	<ol style="list-style-type: none"> <li>1. Logged in as doctor or patient.</li> <li>2. Doctor is on planner screen.</li> <li>3. Network connection is available</li> </ol>
Postconditions	<ol style="list-style-type: none"> <li>1. The application displays events to the user.</li> </ol>

Table 4.10: Check Diary Use Case Specification

#### 4.1.11: Complete Use Case

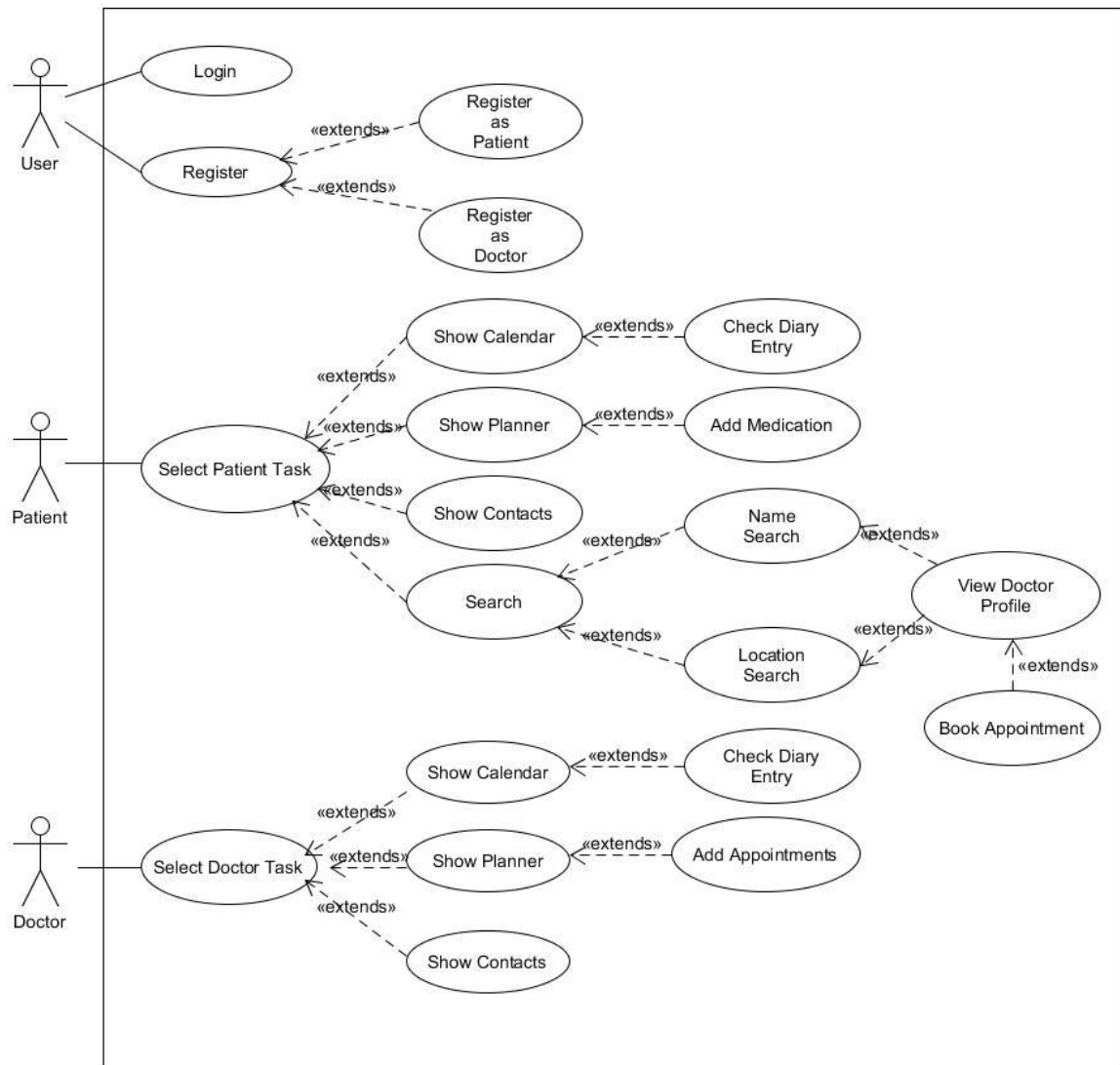


Figure 4.11: Complete Use Case

## 4.2: Activity Diagram

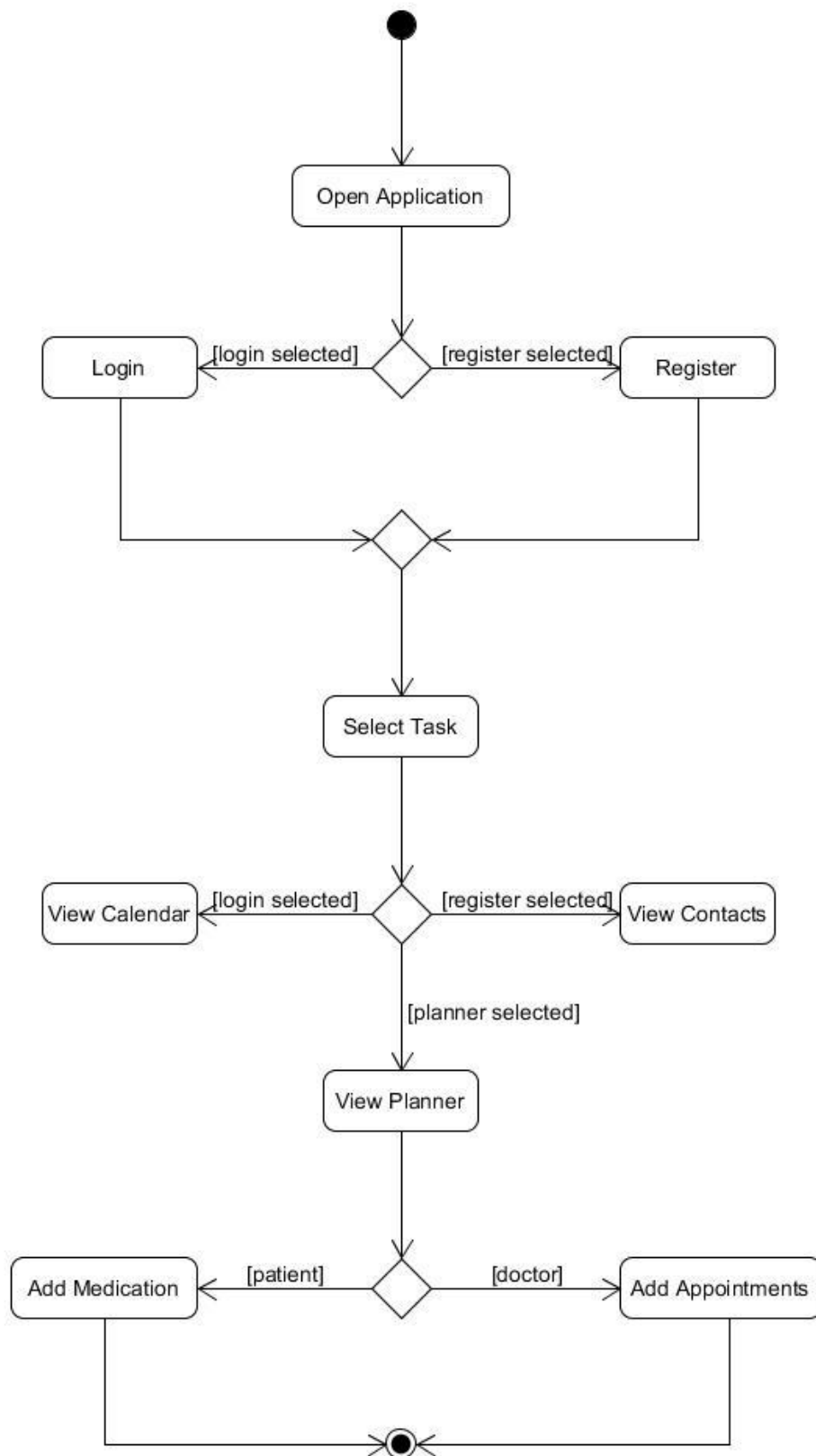


Figure 4.12: Activity Diagram

### 4.3: Data Flow Diagram

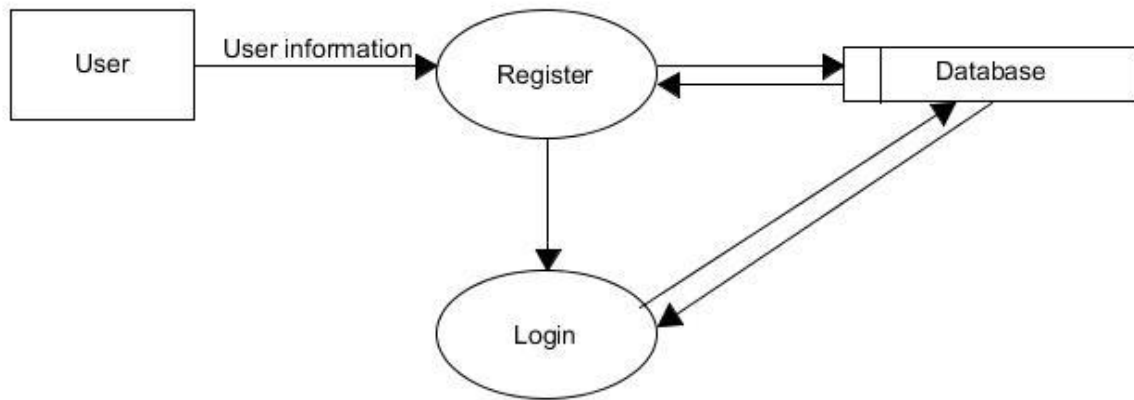


Figure 4.13: Data Flow Diagram

#### 4.4: Sequence Diagrams

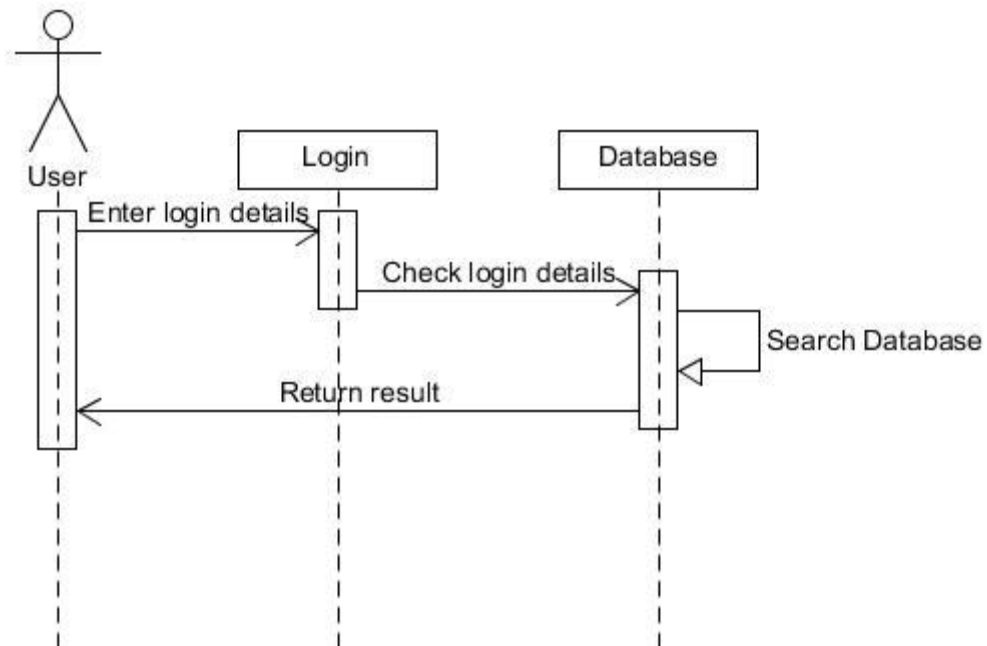


Figure 4.14: Login Sequence Diagram

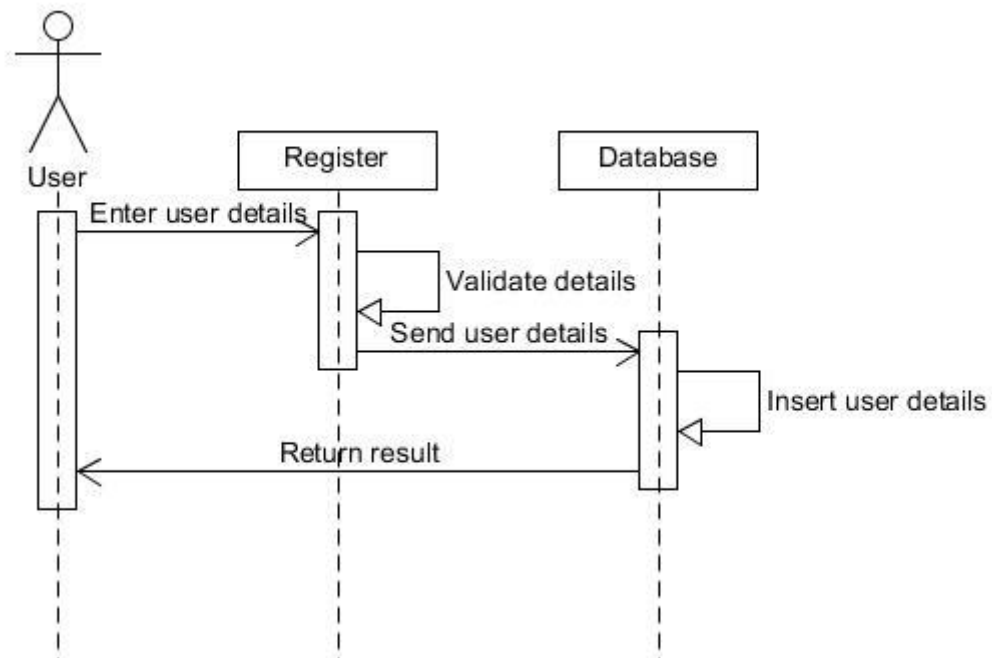


Figure 4.15: Register Sequence Diagram

# Chapter 5: System Design

## 5.1: Application Design

### 5.1.1.1: Navigation

One of the most important consideration when designing the application was navigation. The users should know where they are at all times and how to get to the places they want easily. The action bar will be designed with the intention of helping the users to navigate the application. The action bar will display the current activity the user is in. When the user selects an activity which is a child of another activity, the action bar will provide the user with an up button which will navigate them back to the parent activity when selected.

### 5.1.2: Scaling

The application has been designed with scaling in mind. In order for the application to run effectively on different types of Android devices of all different sizes the layouts must be as flexible as possible. The application will try to stay away from using fixed values for sizing components but if it does it will not use pixels, instead it will use device independent pixels. It will use device independent pixels because when the size is defined with device independent pixels it will not change as it will be compatible with devices with different screen densities. The application will use scale independent pixels for font sizes when the font sizes need to be defined in plain text. Scale independent pixels are very similar to dips, the only difference is that they are affected by the font size defined in system settings. Besides for text, the application will also have all its images saved in different sizes for the different screen sizes. The images will be placed in the appropriate screen size folder based on their size.

## 5.2: User Interface and Functional Design

### 5.2.1: Welcome Activity

The “Welcome” activity is the first activity to open when the application is launched. It checks if the user is logged in by checking the shared preferences. If the user is logged in they get send to the “Home” activity. If they are not logged in they can select the login or register button.

#### 5.2.1.1: User Interface Requirements

- ImageView containing the application logo.
- Button to login.
- Button to register.





Figure 5.1: Welcome activity design

### 5.2.2: Login Activity

The “Login” activity allows users to sign in to the application. The user enters in their login information, it is then validated and verified. If their login details are correct they are sent to the “Home” activity and their information is stored in shared preferences. If their login details are incorrect they are shown a toast message telling them their login information is incorrect.

#### 5.2.2.1: User Interface Requirements

- Two EditTexts for username and password.
- Button to login.

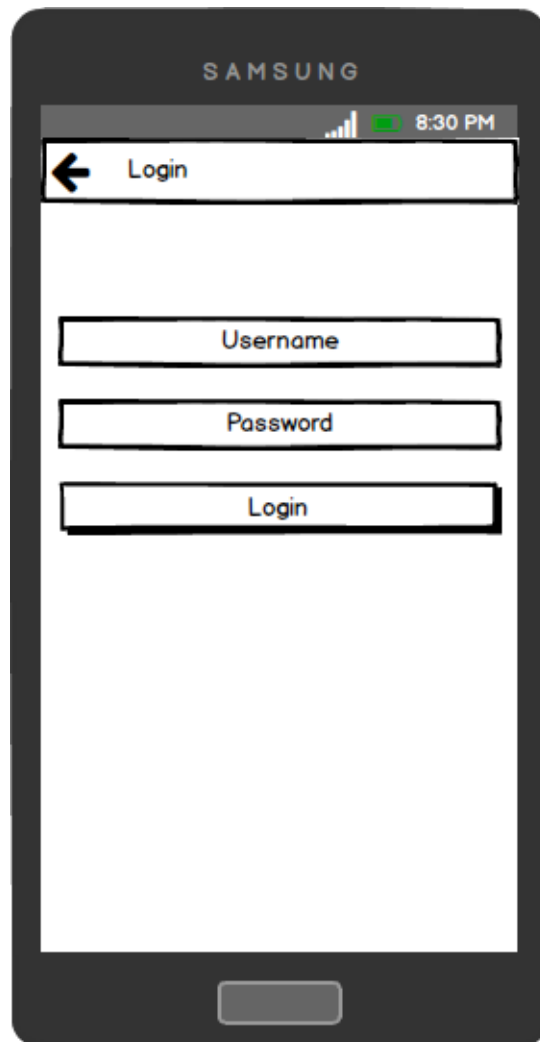


Figure 5.2: Login activity design

### 5.2.3: Register Activity

The “Register” activity allows users to choose to register as a patient or as a doctor. If they select the “Register as Patient” button they are sent to the “Register as Patient” activity or if they select the “Register as Doctor” button they are sent to the “Register as Doctor” activity.

#### 5.2.3.1: User Interface Requirements

- Button to register as a patient.
- Button to register as a doctor.



Figure 5.3: Register activity design

#### 5.2.4: Patient Register Activity

The “Patient Register” activity allows users to register as patients. The user can enter in their information and submit it. The information is then validated and send to the database. If it is successfully stored in the database they are sent to the “Home” activity and their information is stored in shared preferences. If something goes wrong a toast message is displayed to the user letting them know there is a problem.

##### 5.2.4.1: User Interface Requirements

- Seven EditTexts for user’s first name, last name, username, password, address line, city, and county.
- Spinner for countries available to select.
- Button to register as a patient.

The image shows a mobile application interface for patient registration. It features a series of input fields for personal and contact information, followed by a dropdown menu for country selection, and a final registration button. The interface is presented within a Samsung smartphone frame.

Figure 5.4: Patient Register activity design

### 5.2.5: Doctor Register Activity

The “Doctor Register” activity allows users to register as doctors. The user can enter in their information and submit it. The information is then validated and send to the database. If it is successfully stored in the database they are sent to the “Home” activity and their information is stored in shared preferences. If something goes wrong a toast message is displayed to the user letting them know there is a problem.

#### 5.2.5.1: User Interface Requirements

- Eleven EditTexts for user’s first name, last name, username, password, background, qualifications, years’ experience, address line, city, and county.
- Spinner for countries available to select.
- Button to register as a doctor.

The image shows a mobile application interface for a "Doctor Register" activity. The screen is framed by a dark grey border representing the phone's bezel. At the top, the status bar displays "SAMSUNG" in the center, signal strength bars on the left, a battery icon and "8:30 PM" on the right. Below the status bar is a white header bar with a black back arrow on the left and the text "Doctor Register" in the center. The main content area contains eleven input fields stacked vertically: "First Name", "Last Name", "Username", "Password", "Background", "Qualifications", "Years Experience" (with a small text box to its right), "Address line 1, 2", "City", "County, State, Province", and "Country" (a spinner with a downward arrow). At the bottom of the form is a grey button with the text "Register".

Figure 5.5: Doctor Register activity design

### 5.2.6: Home Activity

The “Home” activity allows the users to manage their accounts. The activity will display three fragments as tabs that can be swiped. The activity checks the users shared preferences to get the user type. If the user is a patient they will have a “Search” button in their action bar and will be able to search for doctors by pressing it. The user can also choose to log out by selecting the log out button in the action bar.

#### 5.2.6.1: User Interface Requirements

- Three fragments Calendar, Planner, and Contacts.
- Button to log out.
- If user type is patient: Button to search.



Figure 5.6: Home activity design

### 5.2.7: Search Activity

The “Search” activity allows patients to select a search method to search for doctors. If they select the “Search by Name” button they will be sent to the “Search by Name” activity, or if they select the “Search by Location” button they will be sent to the “Search by Location” activity.

#### 5.2.7.1: User Interface Requirements

- Button to search by name.
- Button to search by location.



Figure 5.7: Search activity design

### 5.2.8: Search by Name Activity

The “Search by Name” activity allows the patients to search for doctors based on their names. The patient can enter the doctor’s name and click the “Search” button. The activity will check the database for doctors with similar names and display them in a ListView below.

#### 5.2.8.1: User Interface Requirements

- EditText to enter name.
- Button to start search.
- ListView for results

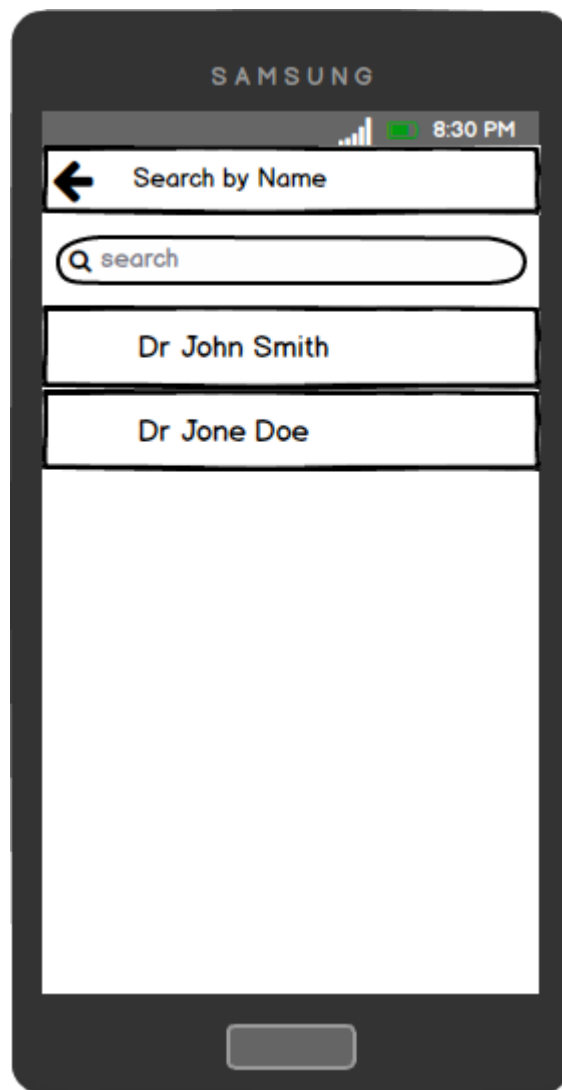


Figure 5.8: Name Search activity design



### 5.2.9: Search by Location Activity

The “Search by Location” activity allows the patients to search for doctors based on location. The patient can select the “Enter Details” button and a dialog will be shown. The patient can then enter in their address manually, or use their saved address, or current location as the address entry. They can then select the max distance and click the “Ok” button. The application will then display all doctors within the max distance of the inputted address to the patient in the map fragment. The doctors will be marked with map makers and a circle will be used to show the max distance radius.

#### 5.2.9.1: User Interface Requirements

- Button to open dialog.
- Dialog to enter details in.
- Two EditTexts for address and max distance.
- Radio group of two radio buttons for determining where to get address from.
- Map Fragment to display map.
- Map Markers to display doctor locations.
- Circle to display distance radius.

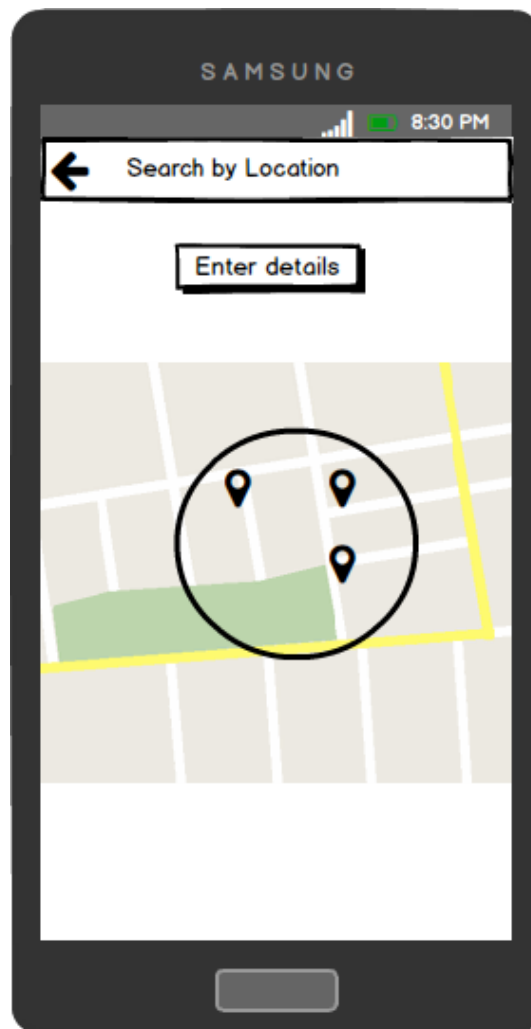


Figure 5.9: Location Search activity design

### 5.2.10: Calendar Fragment

The “Calendar” fragment will contain a CalendarView which allows users to select the dates. If the user selects a date it will pass that date to the “Diary Entry” activity.

#### 5.2.10.1: User Interface Requirements

- CalendarView to select dates.



Figure 5.10: Calendar fragment design

### 5.2.11: Diary Entry Activity

The “Diary Entry” activity allows users to see events for a certain date. The diary event activity receives a date from the “Calendar” fragment. It displays that date at the top of the activity in a TextView. It then checks the database for events on that date. If there are any events it displays them in a ListView below.

#### 5.2.11.1: User Interface Requirements

- TextView to display date.
- ListView to display events.

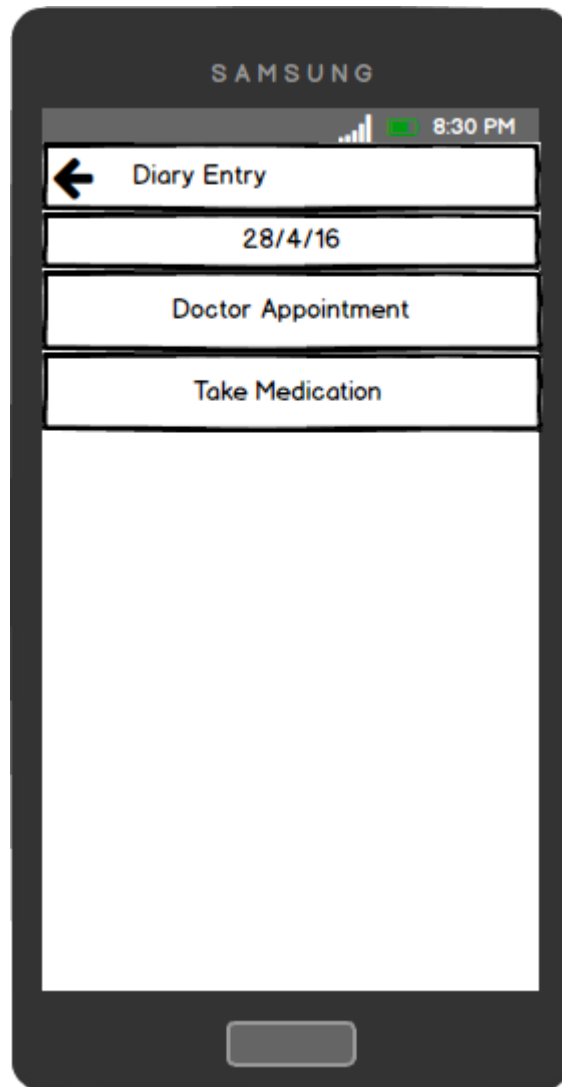


Figure 5.11: Diary Entry activity design

### 5.2.12: Contacts Fragment

The “Contacts” fragment allows users to see their contacts. The contact activity checks the database for contact relationships which contain the user’s id. If there are any contacts it displays them in a ListView. If the user selects a contact, the application will call the contact.

#### 5.2.12.1: User Interface Requirements

- ListView to display contacts.

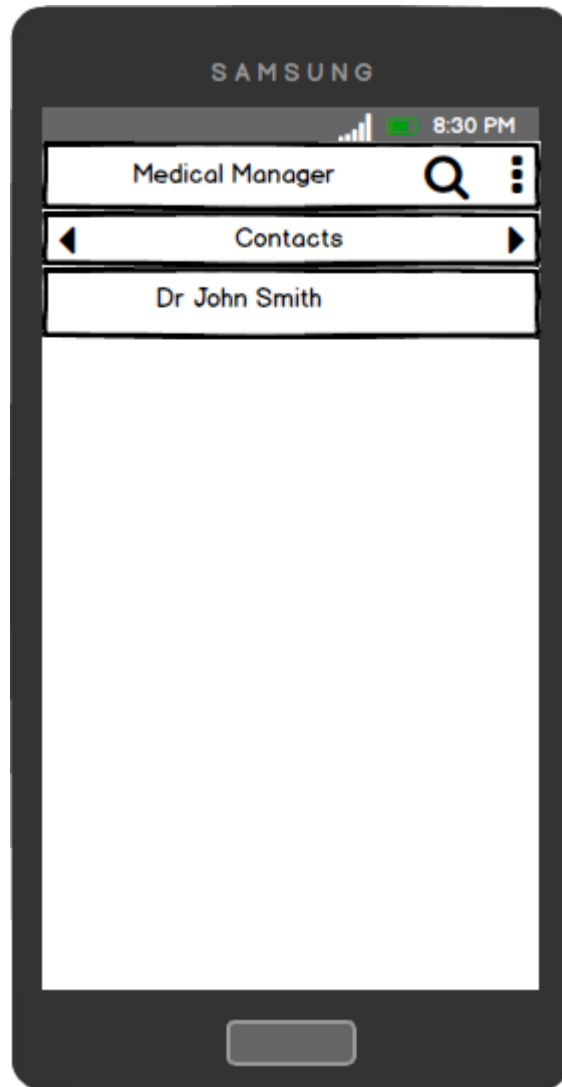


Figure 5.12: Contacts fragment design

### 5.2.13: Planner Fragment

The “Planner” fragment allows users to add schedules. The planner fragment checks the user’s shared preferences to see what type of user they are. If they are a patient they are allowed to add medication schedules and if they are a doctor they are allowed to add appointment schedules. If they select the add button as a patient they will be sent to the “Add Medication” activity, and if they select it as a doctor they will be sent to the “Add Appointments” activity.

#### 5.2.13.1: User Interface Requirements

- If user type is patient: Button to add medications
- If user type is doctor: Button to add appointments

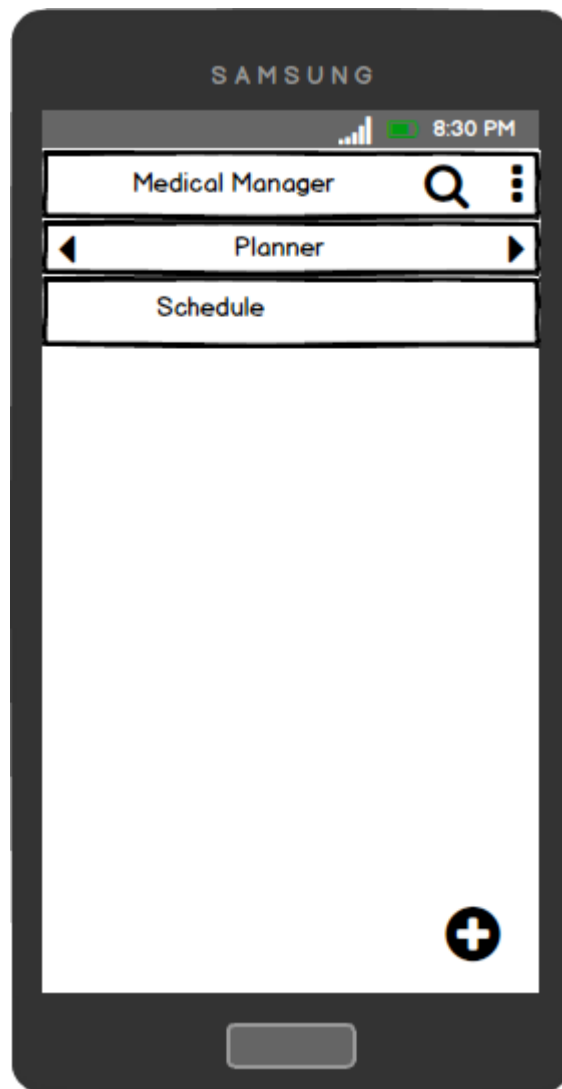


Figure 5.13: Planner fragment design

### 5.2.14: Add Medication Activity

The “Add Medication” activity allows patients to add a medication schedule. The patient can enter in the medication name, start date, time and duration they want to take that medication. When the patient selects the “Start Date” button they will be shown a date picker dialog where they can select a date. When the patient selects the “Time” button they will be shown a time dialog where they can select a time. If the doctor selects the “Number of Days” radio button they will be shown a number picker dialog where they can select the amount of days they want as the duration. When the patient selects the “Done” button their schedule will be saved to the database and they will be reminded from that date on, at that time, for that amount of days to take that medication.

#### 5.2.14.1: User Interface Requirements

- EditText for medication name
- Button to display date picker dialog.
- Date picker dialog to select date.
- Button to show time dialog.
- Time dialog to select time.
- Radio group to determine duration.
- Number picker dialog to select number of days.

The image shows a mobile application interface for adding medication. The screen is titled "Add Medications" with a back arrow. It contains the following elements:

- Medication Name:** A text input field.
- Start Date:** A date picker showing "2016-4-28".
- Time:** A time picker showing "10:00:00".
- Duration:** Two radio buttons: "Continuous" and "Number of days".
- Done:** A button at the bottom of the form.

Figure 5.14: Add Medication activity design

### 5.2.15: Add Appointments Activity

The “Add Appointments” activity allows doctors to add an appointment schedule. The doctor can enter in the schedule name, start date, start time, duration of each appointment, amount of appointments per day and the duration of the schedule. When the doctor selects the “Start Date” button they will be shown a date picker dialog where they can select a date. When the doctor selects the “Start Time” button they will be shown a time dialog where they can select a time. When the doctor selects the “Duration (Mins)” button they will be shown a number picker dialog where they can select a number as the duration of each appointment. When the doctor selects the “Amount” button they will be shown another number dialog where they can select a number for the amount of appointments per day. If the doctor selects the “Number of Days” radio button they will be shown a number picker dialog where they can select the amount of days they want as the schedule duration. When the doctor selects the “Done” button their appointment schedule will be saved to the database and patients will then be able to book those appointments.

#### 5.2.15.1: User Interface Requirements

- EditText for medication name
- Button to display date picker dialog.
- Date picker dialog to select date.
- Button to show time dialog.
- Time dialog to select time.
- Three Buttons to show number picker
- Three Number picker dialogs to select numbers

The screenshot shows a mobile application interface for adding appointments. At the top, there's a status bar with 'SAMSUNG' and '8:30 PM'. Below it is a title bar with a back arrow and 'Add Appointments'. The main form contains the following elements:

- Appointment Schedule Name:** A text input field.
- Start Date:** A date picker showing '2016-4-28'.
- Start Time:** A time picker showing '10:00:00'.
- Duration (Mins):** A number picker showing '15'.
- Amount:** A number picker showing '1'.
- Duration:** Two radio buttons, 'Continuous' (selected) and 'Number of days'.
- Done:** A button at the bottom of the form.

Figure 5.15: Add Appointments activity design

### 5.2.16: Profile Activity

The “Profile” activity allows patients to see doctors’ profiles. They can see information about the doctors such as their name, background, qualifications, years’ experience, and address. The patient can select the “Add as Contact” button to add the doctor as a contact. The patient can also check if the doctor has any available appointment slots by pressing the “Check Available Appointments” button. This will send them to the “Available Appointments” activity.

#### 5.2.16.1: User Interface Requirements

- Five TextViews for doctor name, background, qualifications, years’ experience, and address.
- Button to add as contact
- Button to check available appointments

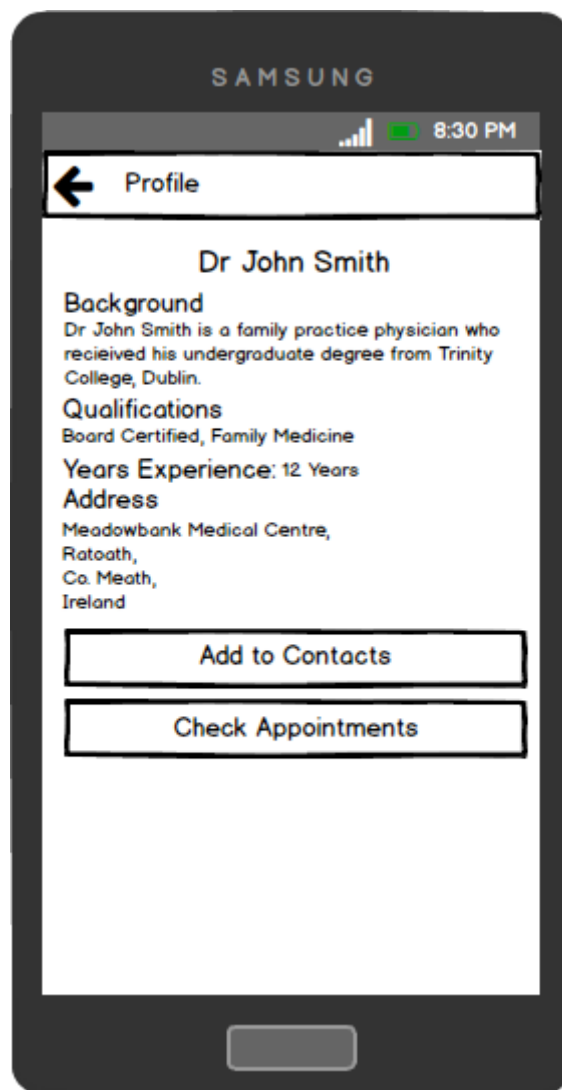


Figure 5.16: Profile activity design



### 5.2.17: Available Appointments Activity

The “Available Appointments” activity allows patients to see available appointments for doctors. If the patient selects a date on the CalendarView it will display the date in a TextView below the CalendarView. The activity will then check the database to see if there are any available appointments on that date. If there are any it will display them in a ListView below. The patient can then select and book an appointment.

#### 5.2.17.1: User Interface Requirements

- CalendarView to select dates.
- TextView to display selected date.
- ListView to display appointments on selected date.

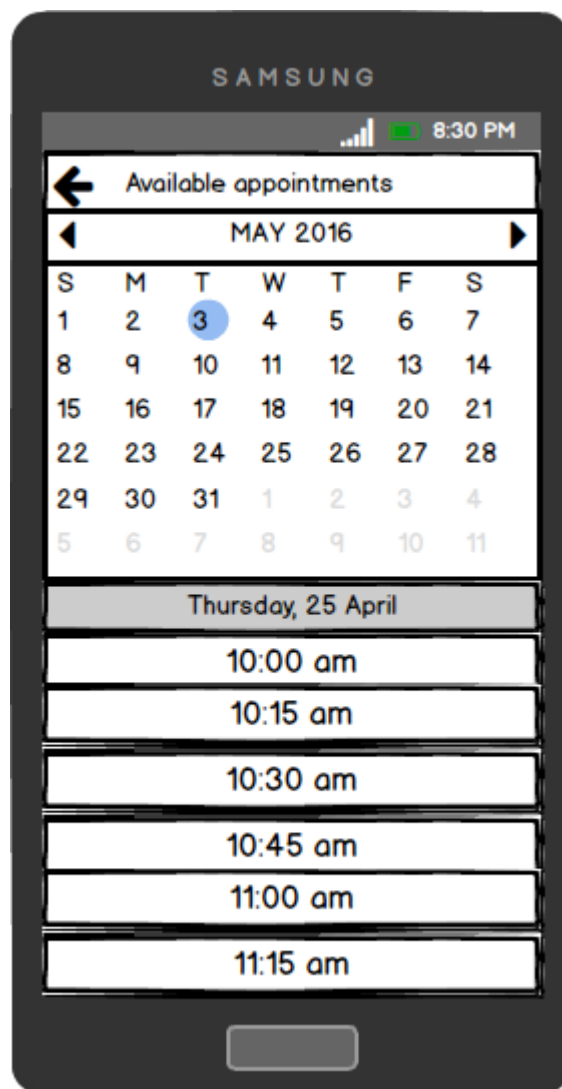


Figure 5.17: Available Appointments activity design

## 5.3: Class Diagram

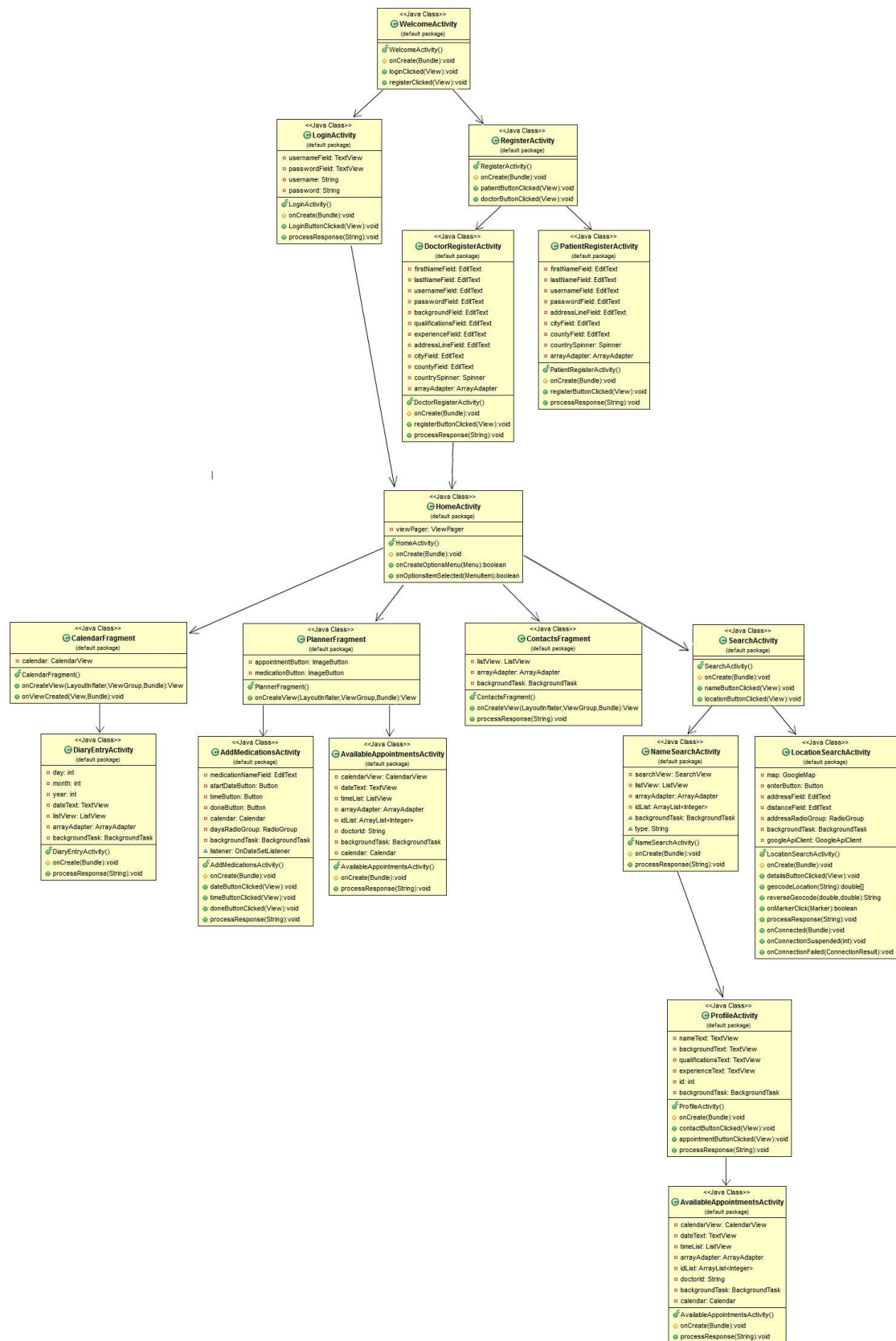


Figure 5.18: Class Diagram

## 5.4: Entity Relationship Diagram

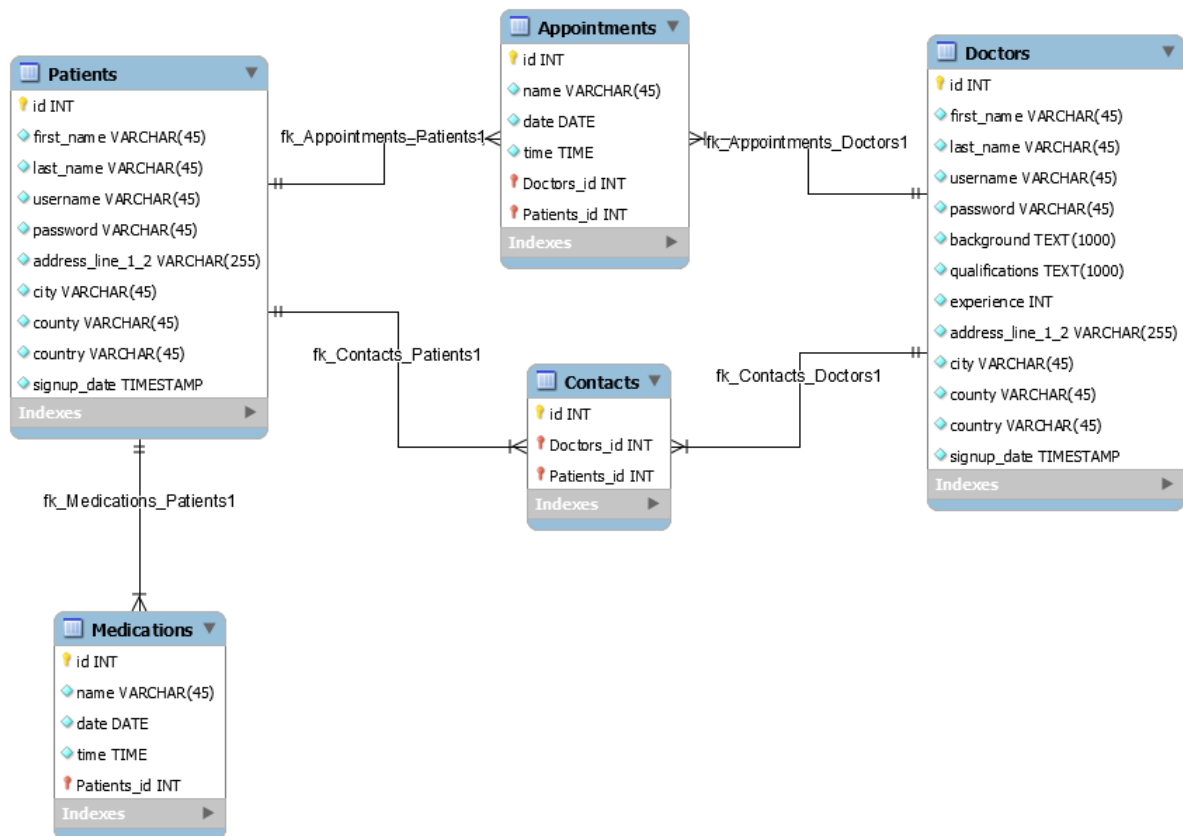


Figure 5.19: Entity Relationship Diagram

## Chapter 6: Implementation of Prototype

For the most part, the development of the application remained true to the design specifications outlined. The application was developed using Java, MySQL, PHP and JSON. Java was used to code the application in Android Studio using the Android SDK. A MySQL database was used to store the users' data remotely. PHP was used to communication between the application and the database. JSON was used as the format to transfer data between the application and database.

### 6.1: User Information Class

This class was implemented to allow the application to store the user's information locally on the device. It uses the Shared Preferences class to create a file on the user's device. When the save method is called it updates the shared preferences. The class also contains a number of getter methods which allow other activities to get specific user information.

### 6.2: Background Task class

The "Background Task" class was implemented to provide long tasks such as database requests a worker thread to work on, rather than having them running on the UI thread and freezing the application. The background task extends the AysncTask class. The background class handles multiple database requests which it controls with a switch statement. When another activity wants to make a database request they will need to pass through the type of request and necessary parameters for the request.

### 6.3: Welcome Activity

The "Welcome" activity is the first activity displayed to the users. The activity will call the "isLoggedIn" method from the "User Information" class. If "isLoggedIn" is equal to true, the user is logged in a will not have to log in again. If it is false the Welcome activity will wait till one of its button listeners are triggered. If the user selects the "Login" button the "Login" activity will start or if the user selects the "Register" button the "Register" activity will start.

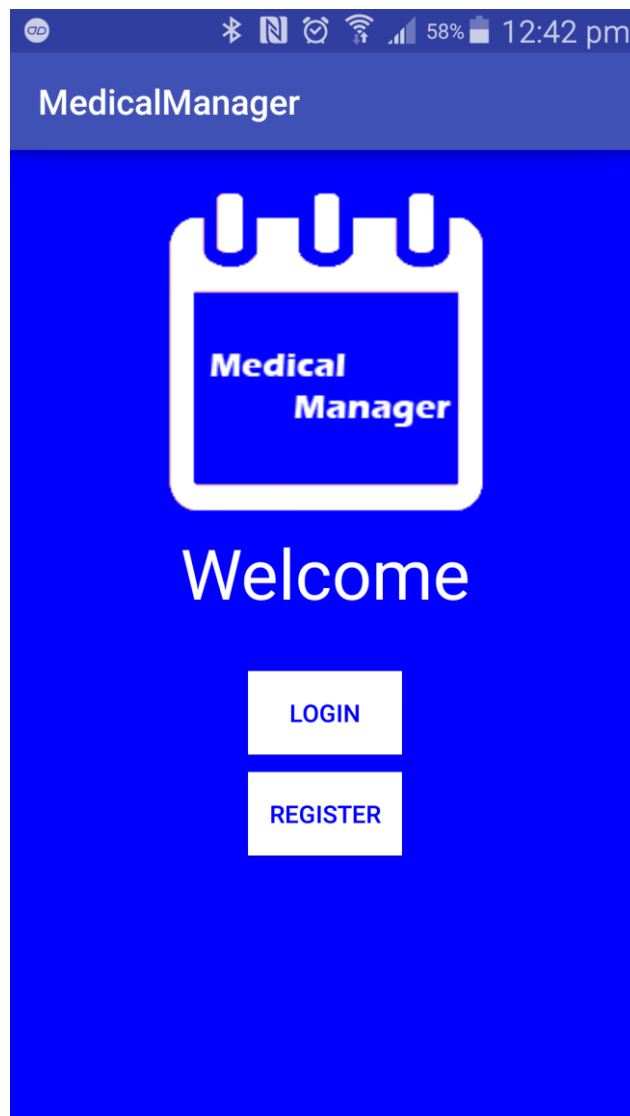


Figure 6.1: Welcome activity implementation

## 6.4: Login Activity

When the user enters in their login details and presses the “Login” button, the login activity will get the text from the fields, convert them to strings. Then the activity will send them to the “Background Task” to be checked in the database. The “Background Task” will return a result to the “Login” activity. The “Login” activity will process the results. If the result says “Login successful” the activity will start the “Home” activity. If it returns a different result it will just display the result to the user through a toast message to let them know what went wrong.

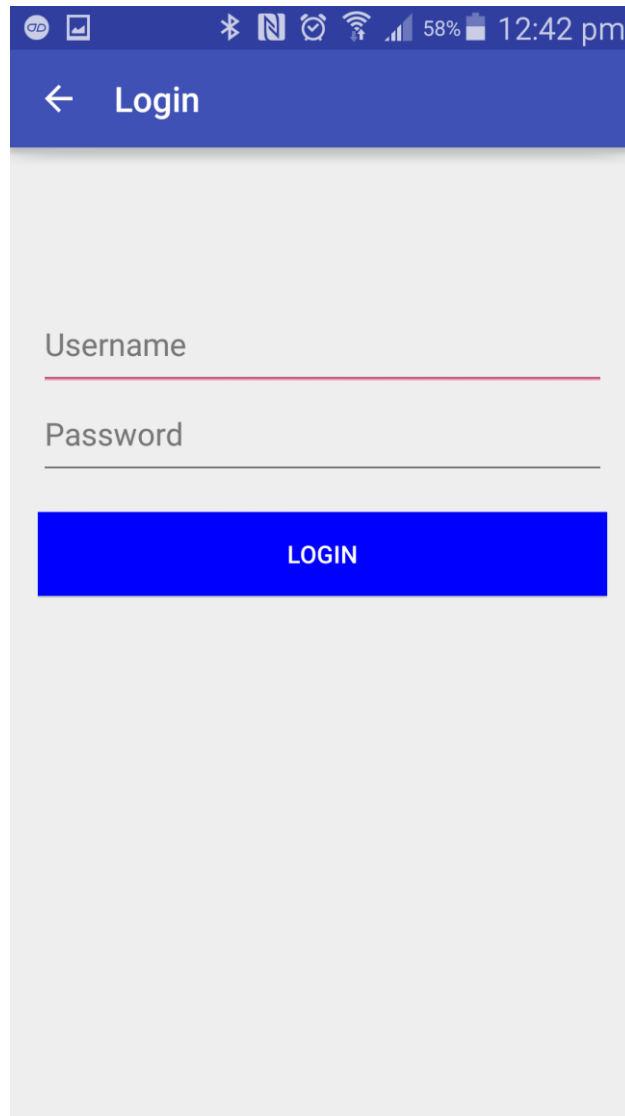


Figure 6.2: Login activity implementation

## 6.5: Register

The “Register” activity has two buttons which have on click listeners. If the “As Patient?” button is selected the activity starts the “Register Patient” activity or if the “As Doctor?” button is selected the activity starts the “Register Doctor” activity.

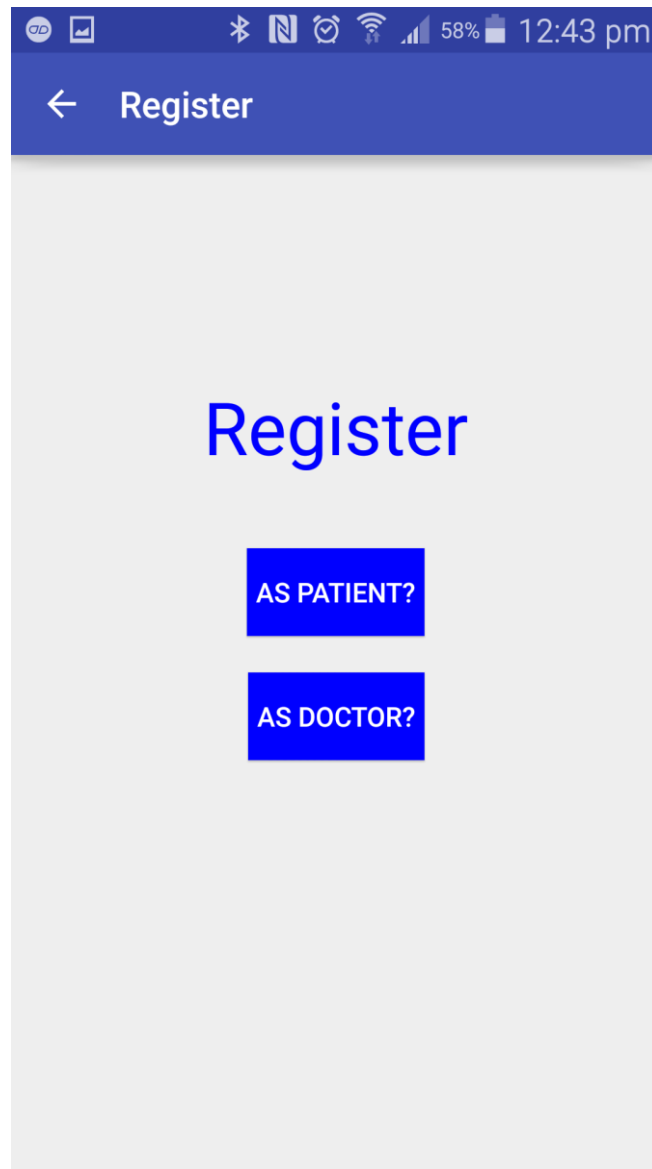


Figure 6.3: Register activity implementation

## 6.6: Register as Patient

The “Register as Patient” activity is started when the user chooses to register as a patient. When the user fills out the fields and presses the “Done” button the activity checks to see if the information is valid. If there are missing fields or the fields don’t match the regular expressions a toast message is displayed telling the user to fix their mistakes. If the information is validated successfully the activity will pass the information to the “Background Task” which will attempt to insert the information into the database. It will then return a message on the status. If it was successful the activity will start the “Home” activity and finish itself.

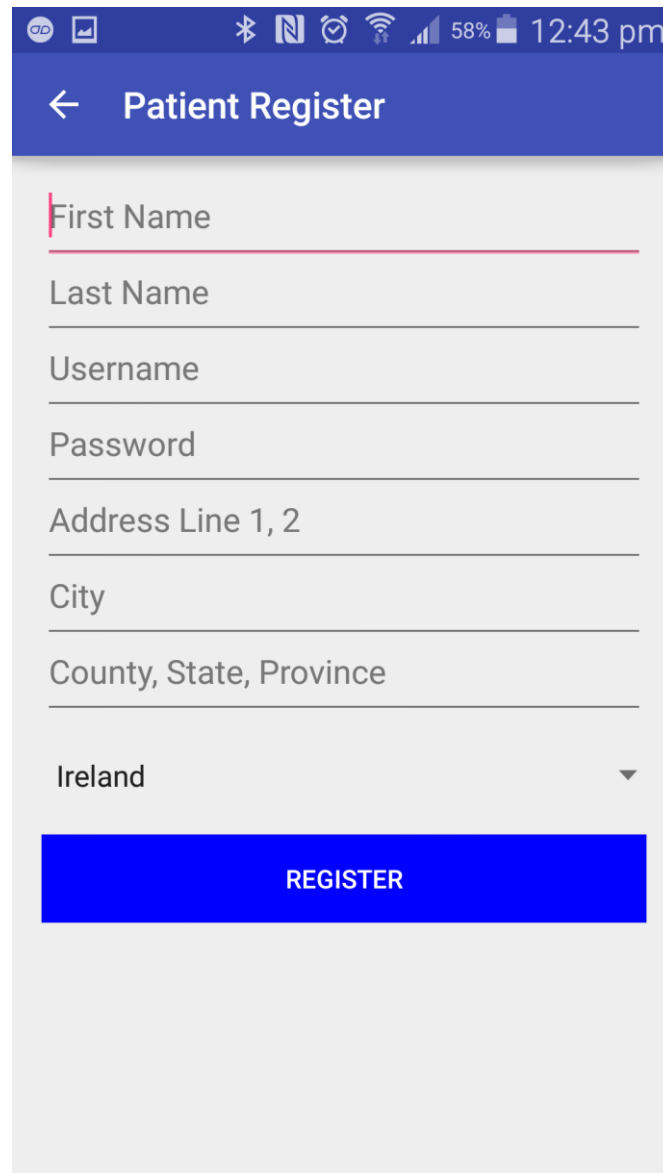
A screenshot of a mobile application interface for a "Patient Register" activity. The screen has a blue header bar with a back arrow and the title "Patient Register". Below the header, there are several text input fields: "First Name", "Last Name", "Username", "Password", "Address Line 1, 2", "City", and "County, State, Province". The "First Name" field is currently active, indicated by a red cursor. Below these fields is a dropdown menu showing "Ireland" with a downward arrow. At the bottom of the form is a large blue button with the text "REGISTER" in white capital letters. The status bar at the top of the phone shows various icons including signal, Wi-Fi, battery (58%), and time (12:43 pm).

Figure 6.4: Patient Register activity implementation



## 6.7: Register as Doctor

The “Register as Doctor” activity is started when the user chooses to register as a doctor. When the user fills out the fields and presses the “Done” button the activity checks to see if the information is valid. If there are missing fields or the fields don’t match the regular expressions a toast message is displayed telling the user to fix their mistakes. If the information is validated successfully the activity will pass the information to the “Background Task” which will attempt to insert the information into the database. It will then return a message on the status. If it was successful the activity will start the “Home” activity and finish itself.

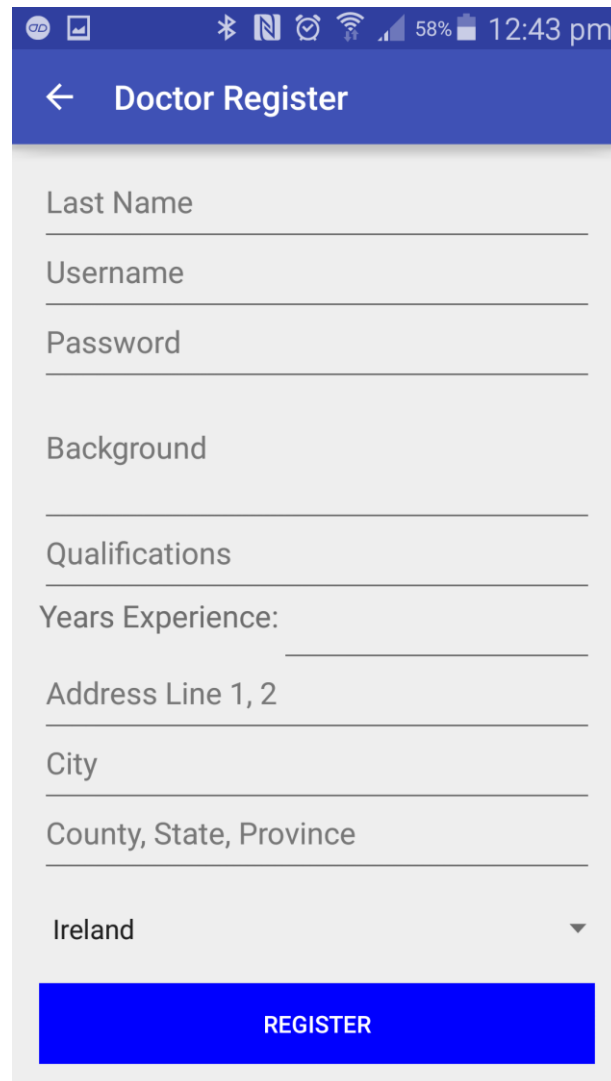
A screenshot of a mobile application interface for registering a doctor. The screen has a blue header bar with a back arrow and the title "Doctor Register". Below the header, there are several text input fields: "Last Name", "Username", "Password", "Background", "Qualifications", "Years Experience:", "Address Line 1, 2", "City", and "County, State, Province". Below these fields is a dropdown menu currently showing "Ireland". At the bottom of the form is a large blue button with the text "REGISTER" in white capital letters. The top of the screen shows a status bar with various icons and the time "12:43 pm".

Figure 6.5: Doctor Patient activity implementation

The “Calendar” fragment has a `CalendarView` which has an on date change listener. When the user selects a date on the calendar it will start a diary entry activity and pass it the date that was selected.



## 6.9: Diary Entry Activity

The “Diary Entry” activity receives the date from the calendar fragment. The activity then gets the username of the user logged in by calling the “getUsername” method from the “User Information” class. The activity then passes the username and date to the “Background Task”. The “Background Task” checks and returns all appointments and medications with the same date and username from the database. The activity then displays the results in the listView.

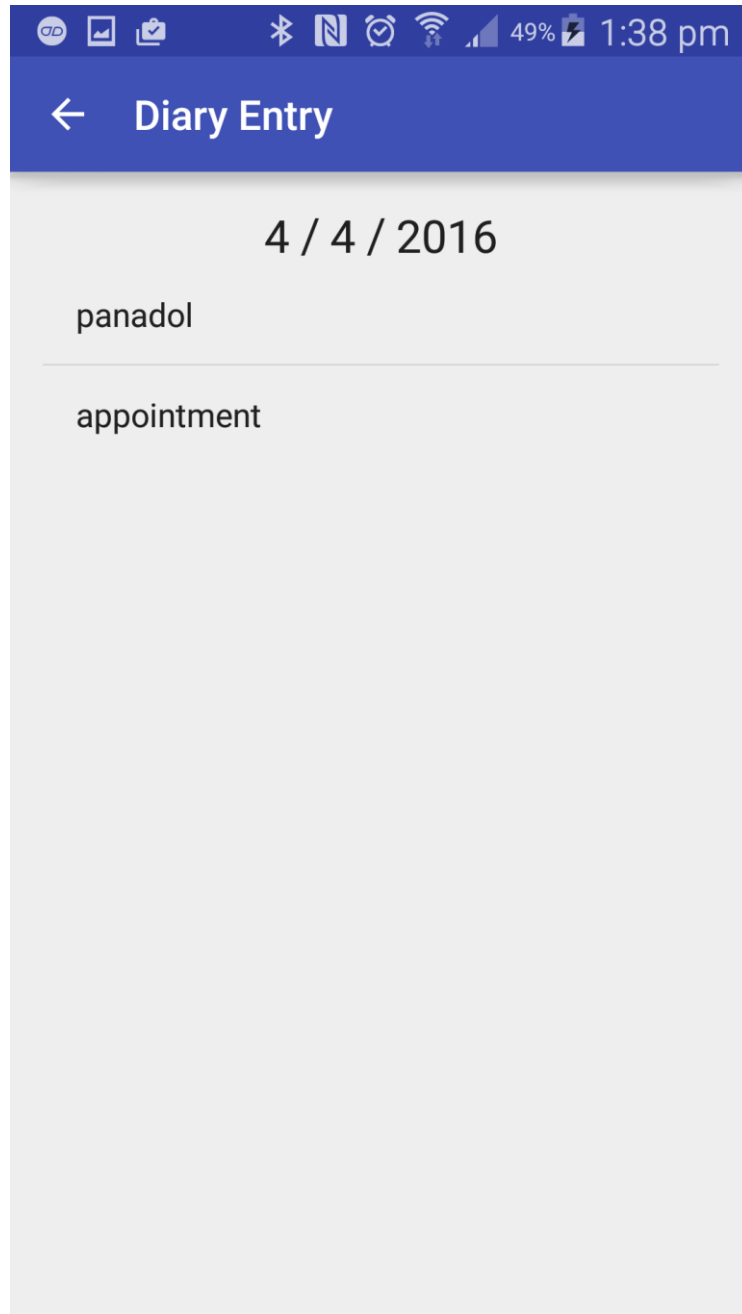


Figure 6.7: Diary Entry activity implementation

## 6.10: Planner fragment

The “Planner” fragment gets the logged in user type from the “User Information” class and then checks if the user is a patient or a doctor. If the user is a patient they will be shown the “Add Medication” button, and if they are doctor they will be shown the “Add Appointments” button. If the user presses the add button logged in as patient the fragment will start the “Add Medication” activity or if they are logged in as a doctor the fragment will start the “Add appointments” button.

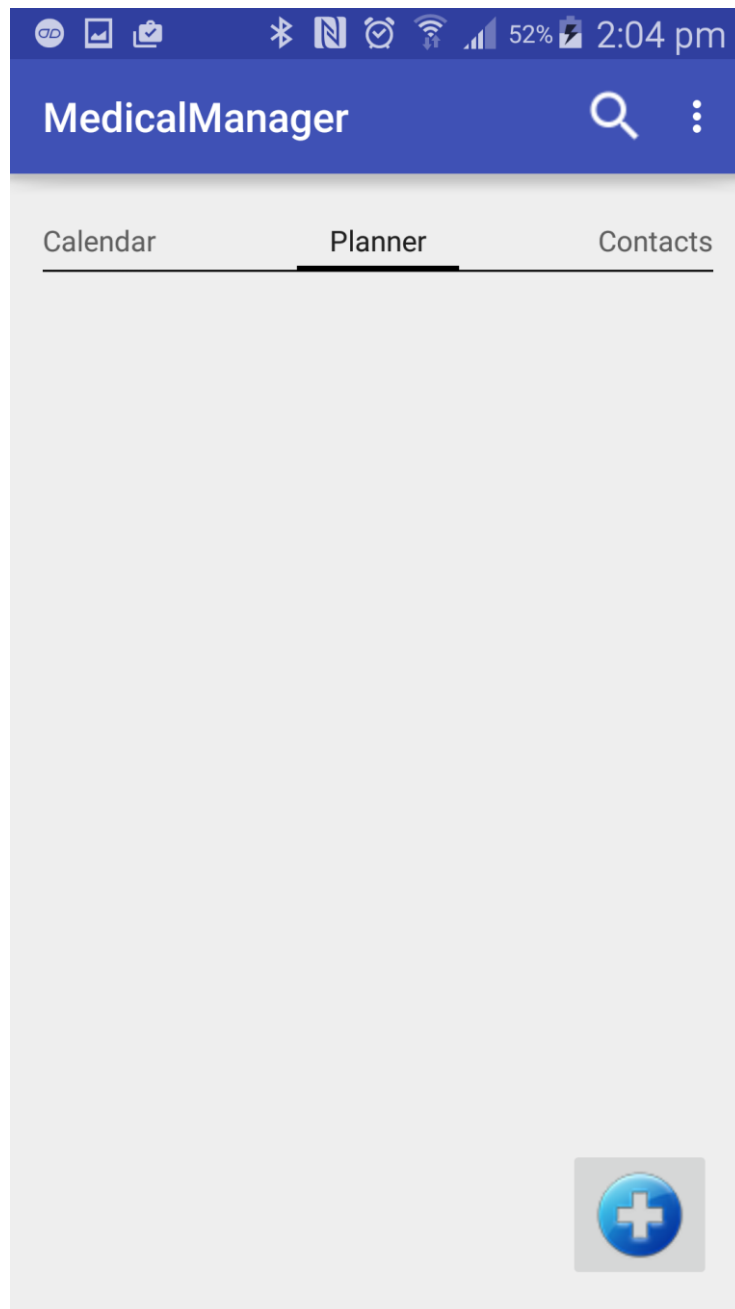


Figure 6.8: Planner fragment implementation

### 6.11: Add Medication activity

When the user enters in the details for their medication schedule and presses the “Done” button, the “Medication” activity will get the user’s username from the “User Information” class and will then send the medication details and username to the “Background Task”. The “Background Task” will attempt to add the information to the database and return a response to the “Medication” activity. If the response is “Medication successfully added” the activity sets the “Alarm Manager” and the activity starts the “Home” activity.

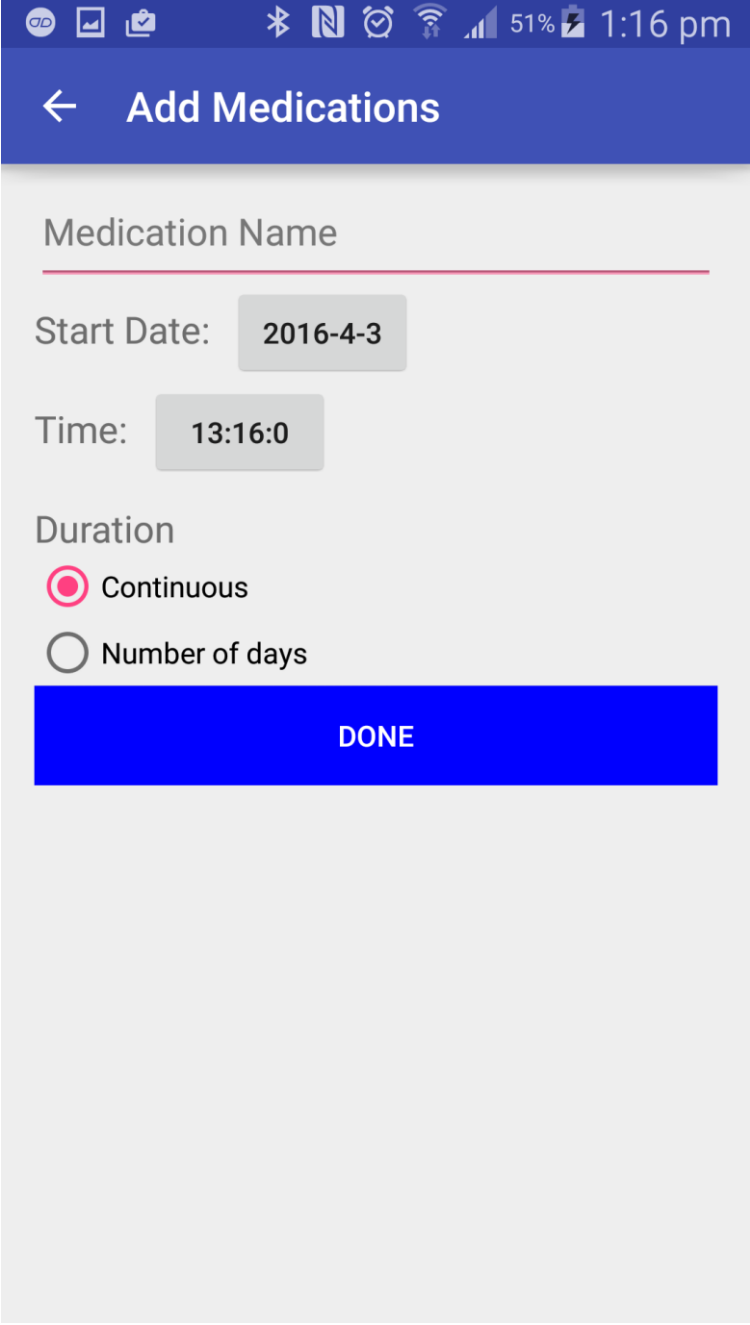
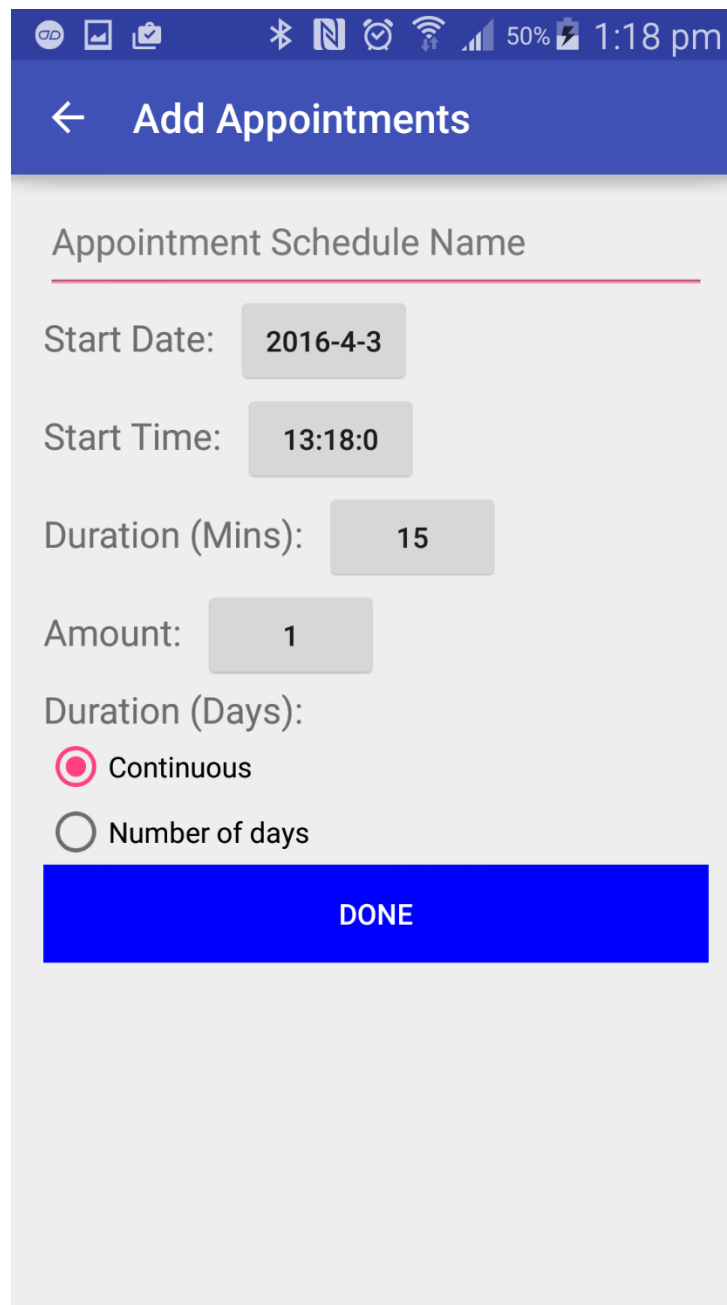
A screenshot of an Android application's 'Add Medications' activity. The screen has a light gray background. At the top is a dark blue header bar with a white back arrow icon on the left and the text 'Add Medications' in white. Below the header, the form contains several fields: 'Medication Name' with a red underline; 'Start Date:' followed by a gray date picker showing '2016-4-3'; 'Time:' followed by a gray time picker showing '13:16:0'; and 'Duration' with two radio button options: 'Continuous' (which is selected with a red dot) and 'Number of days'. At the bottom of the form is a large blue button with the white text 'DONE'. The top of the screen shows a status bar with various icons and the time '1:16 pm'.

Figure 6.9: Add Medication activity implementation

## 6.12: Add Appointments activity

When the user enters in the details for their appointment schedule and presses the “Done” button, the “Appointments” activity will get the user’s username from the “User Information” class and will then send the appointment details and username to the “Background Task”. The “Background Task” will attempt to add the information to the database and return a response to the “Appointment” activity. If the response is “Appointments successfully added” the “Alarm manager” is set and the activity starts the “Home” activity.



The screenshot shows a mobile application interface for adding appointments. At the top, there is a status bar with various icons and the time 1:18 pm. Below this is a blue header bar with a back arrow and the title "Add Appointments". The main content area is light gray and contains several input fields: "Appointment Schedule Name" (with a red underline), "Start Date:" (2016-4-3), "Start Time:" (13:18:0), "Duration (Mins):" (15), and "Amount:" (1). Below these is a section for "Duration (Days):" with two radio button options: "Continuous" (selected) and "Number of days". At the bottom, there is a large blue button labeled "DONE".

Figure 6.10: Add Appointments activity implementation

## 6.13: Contacts

When the “Contacts” fragment is selected, the fragment gets the username from the “User Information” class and then passes the username to the “Background Task” which will return all contacts which the username is associated with. The activity will place them in a listView. If one of the contacts is selected it starts a phone call intent.

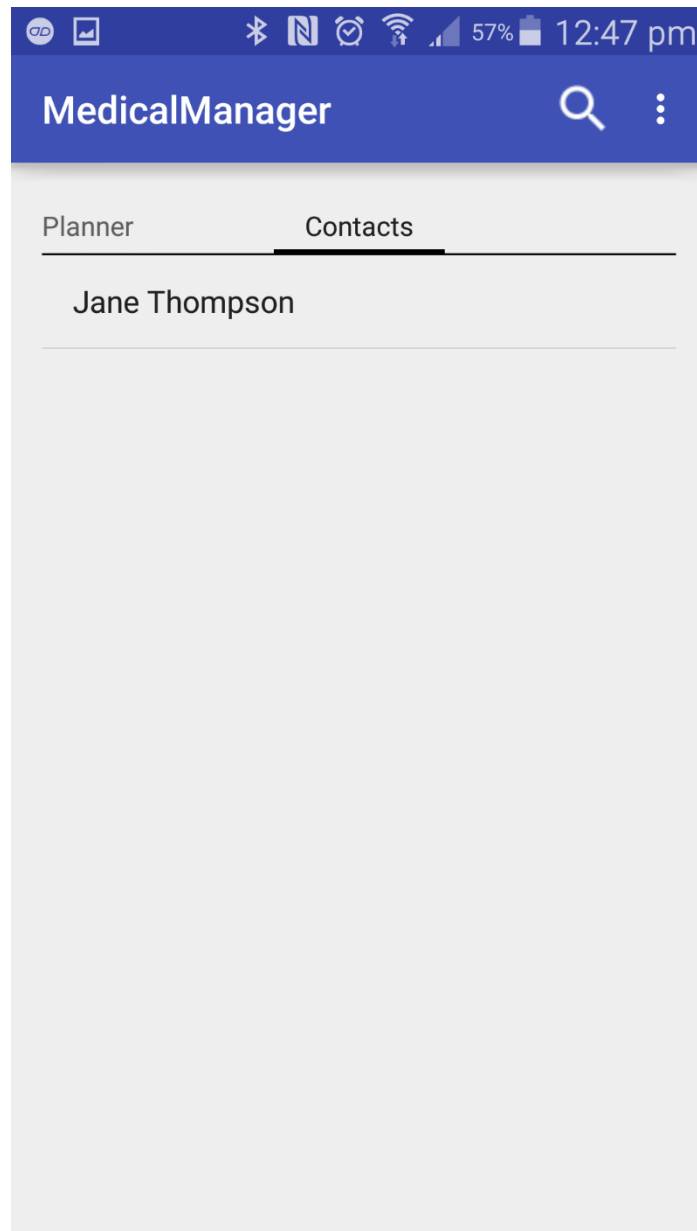


Figure 6.11: Contacts fragment implementation

## 6.14: Search

The “search” activity has two buttons which have on click listeners. If the “Search by Name” button is selected the activity starts the “Search by Name” activity or if the “Search by Location” button is selected the activity starts the “Search by Location” activity.

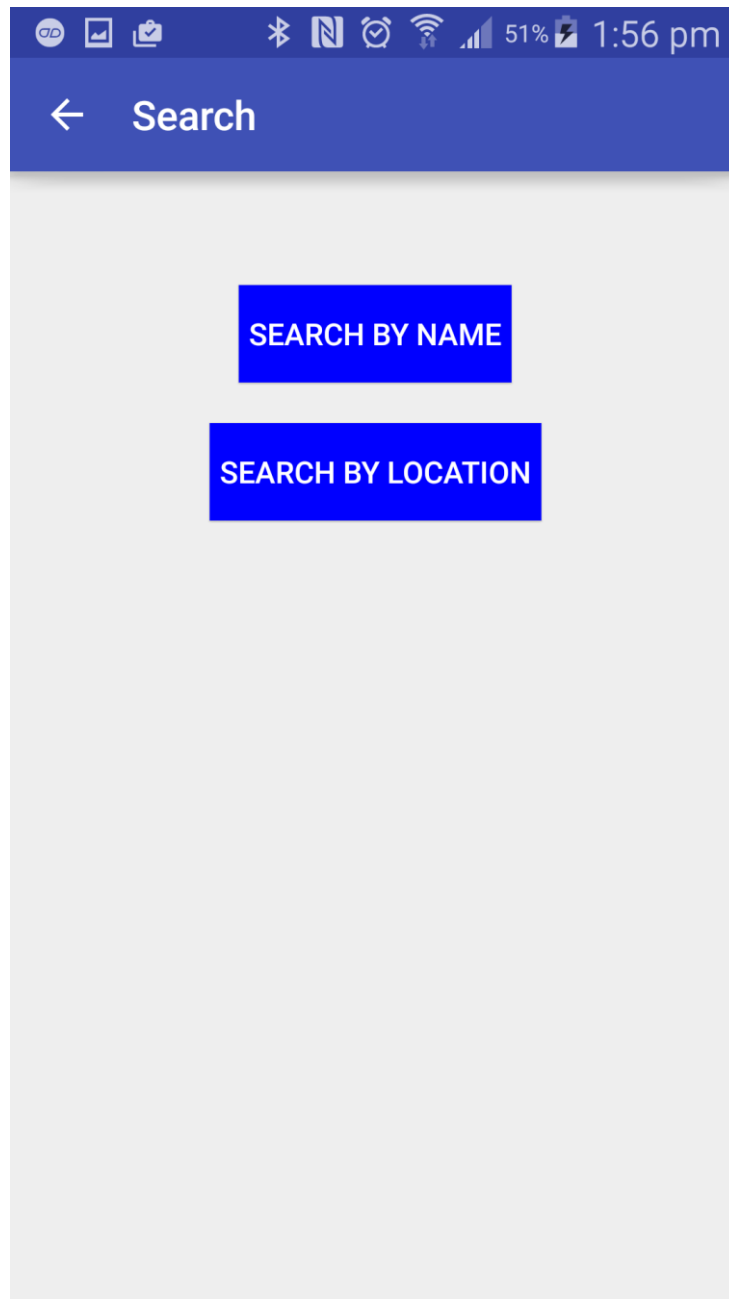


Figure 6.12: Search activity implementation



## 6.15: Search by Name

The “Search by Name” activity has a search view with a set on query text listener. When the user inputs data the query is passed to a “Background task” which searches for doctors with similar names in the database. If any information is returned it is displayed in the list view.

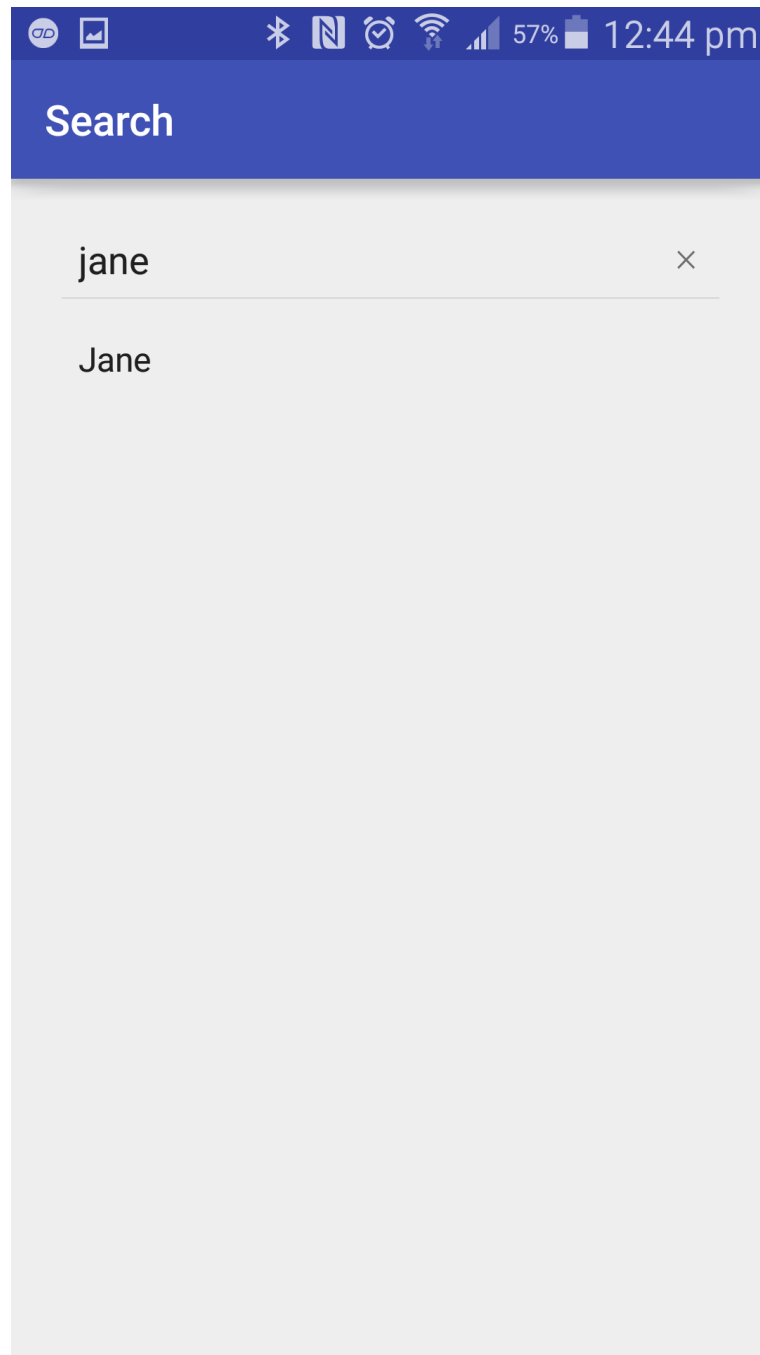


Figure 6.13: Name Search activity implementation

## 6.16: Search by Location

The “Search by Location” activity has a map fragment which is powered by the Google Maps API. There is also an “Enter Details” button which has an on click listener. When the button is selected a custom dialog is displayed. The dialog allows the user to enter in their own address, their account address, or they can even use their current location. They can then enter in their max distance and press the “Ok” button. The map will then show the address the user selected and the doctors available within that distance.

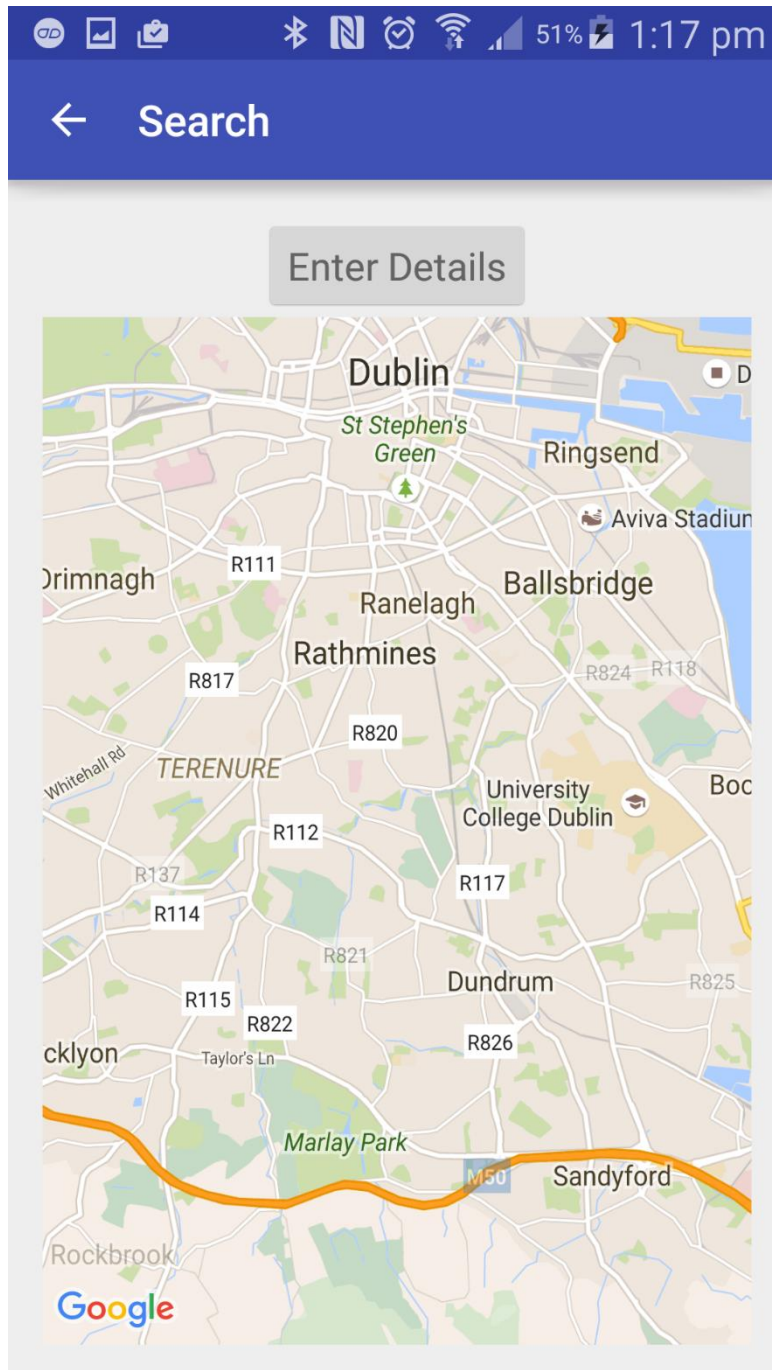


Figure 6.14: Location Search activity implementation

## 6.17: Available Appointments

The “Available Appointments” activity has a CalendarView which has an on date change listener. When the user selects a date on the calendar it will change the textView of the activity to that date. The activity will then pass the profile id and the date to the “Background Task”. The “Background Task” will check if there are any appointments which have the same date and doctor id and nothing entered for the patient id. The “Background Task” will then return the results. The activity will display appointment times in a listview if there are any available times returned. If there are any appointments available and the user selects one the activity will pass the users username and the appointment id to the “Background Task” which will save the users username in the appointment with the selected appointment id so the appointment can no longer be available. The “Alarm manager” is then set to the time and date of the appointment.

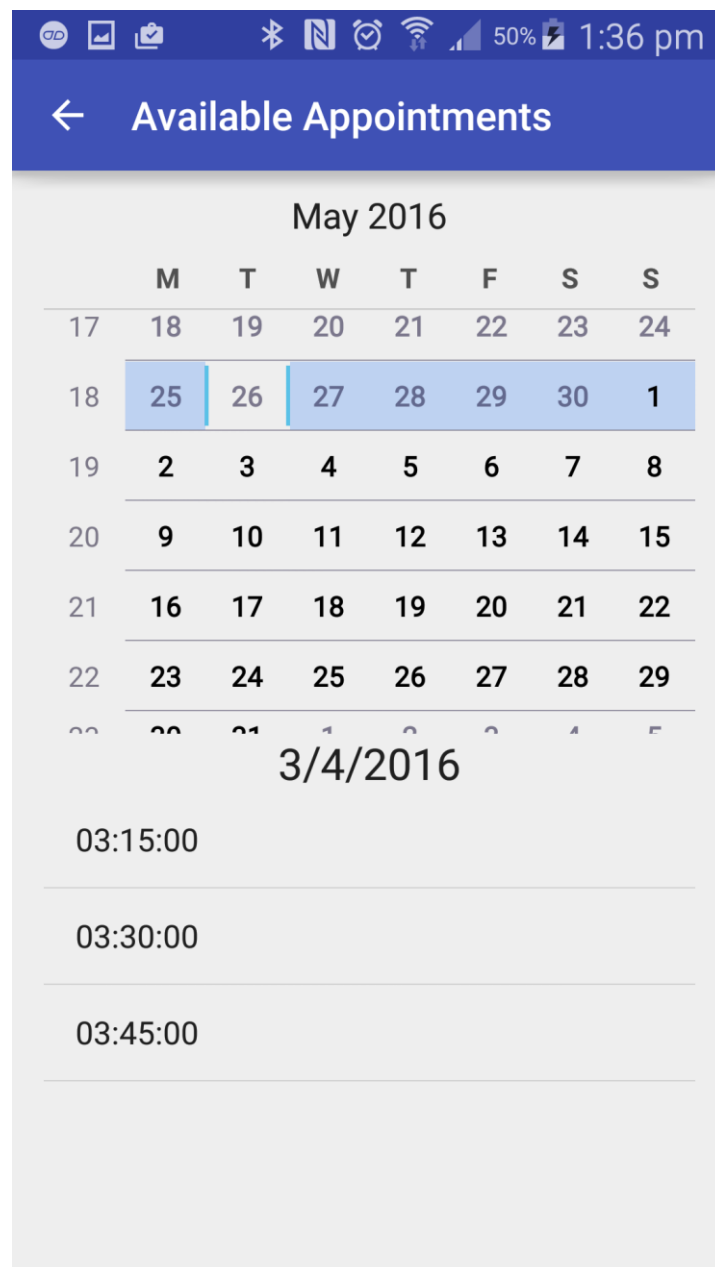


Figure 6.15: Available Appointments activity implementation

## 6.18: Profile

The “Profile” activity displays all of the necessary information about the doctor in text views. There are two buttons with on click listeners. If the “Add as Contact” button is selected the activity will pass the logged in user’s username and the profile id to the “Background Task”. The “Background Task” will try to insert the contact relationship in the database if it does not already exist. If they are already contacts a toast message will be displayed telling the user that they are already contacts. If the patient selects the “Make an Appointment” button the activity will start the “Check Appointments” activity.

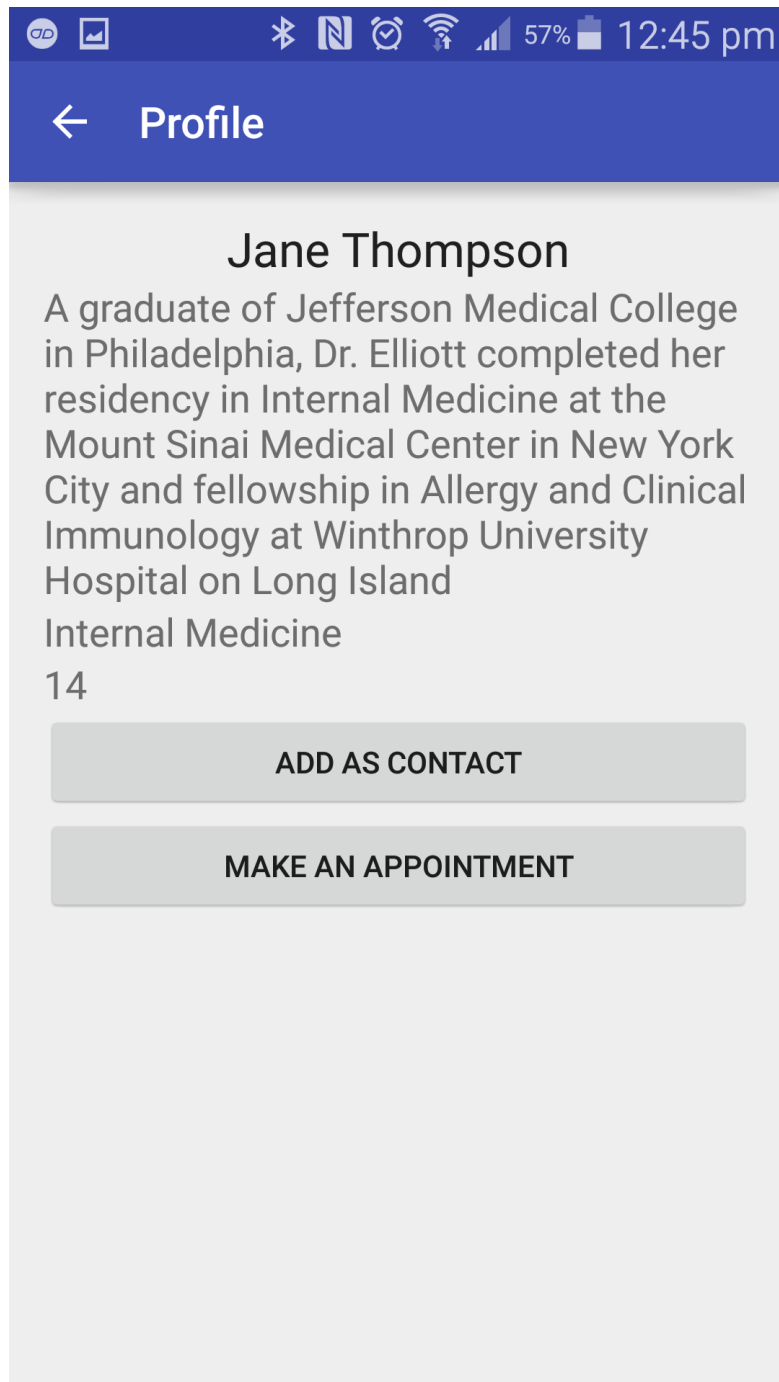


Figure 6.16: Profile activity implementation

# Chapter 7: Testing and Evaluation

## 7.1: Android API Versions

For the development of the application the ANDROID API version remained at 4.0 (Ice Cream Sandwich). Since it uses for Android 4.0 it is compatible with 94% of all android devices. The application has a minimum SDK version of 14 and a target SDK version of 19. The minimum API version of 14 never changed throughout development as the application did not implement any features that required a higher API. The application was tested on the following devices:

- Sony Xperia E – Running on Android 4.0 (Ice Cream Sandwich)
- Samsung S5 – Running on Android 5.0 (Lollipop)
- Nexus 10 – Running on Android 6.0 (Marshmallow)

## 7.2: User Testing and Results

### 7.2.1: Survey

A survey was conducted on one doctor and nine of his patients. The doctor who works at a General Practice was asked to download the application and get nine of his patients to download it. After a week of using the applications the doctor and his patients were asked the following questions shown in Table 7.1.

Question	Good	Bad	Reason
What is your opinion of the user interface?	70	30	Most of the users thought the user interface looked nice and professional.
Where you able to figure out how to use the application without assistance?	65	35	Most users were able to figure out all of the functionality by themselves but some did need assistance.
How does this application compare to other similar applications you have used?	80	20	Most users thought the application was better than the majority of the applications they used, mainly because of features such as medication scheduling and automated appointment booking.
Did the application make it easier for you to communicate?	80	20	Most users were able to communicate with the doctor easier as they had his contact saved.
Did this application help you keep your medical needs more organised?	100	0	All users said that the scheduler and notifications they received helped them to remember important medical events.
Was your overall experience with the application good?	75	25	Most users found at least one of the features beneficial to them.
Would you recommend the application to your friends?	80	20	Most users said they would recommend the application to their friends to help them manage their medical needs.

Table 7.1: Survey

### 7.2.2: Register Test Case

Test Case ID	1
Test Name	Register User
Purpose	To register a new user in the application.
Precondition	The user is not registered.
Assumptions	The user is not registered and is unable to login
Steps	The user enters their information The user clicks the “register” button.
Expected Outcome	The “Home” screen is displayed to user. If register details are invalid display toast saying “Invalid information entered.”
Actual Outcome	As expected.

Table 7.2: Register Test Case

### 7.2.3: Login Test Case

Test Case ID	2
Test Name	Login User
Purpose	To log existing member in to application
Precondition	The user is not logged in.
Assumptions	The user is registered but not logged in yet.
Steps	The user enters their username and password. The user clicks the “login” button.
Expected Outcome	The “Home” screen is displayed to the user. If users login information is invalid display toast saying “Invalid username or password”,
Actual Outcome	As expected.

Table 7.3: Login Test Case

### 7.2.4: Home Test Case

Test Case ID	3
Test Name	Home
Purpose	This test case tests the functionality of the Home Screen
Precondition	The user is logged in.
Assumptions	
Steps	User presses calendar, planner, or contacts tab.
Expected Outcome	The fragment corresponding to the selected tab is returned
Actual Outcome	As expected.

Table 7.4: Home Test Case

### 7.2.5: Calendar Test Case

Test Case ID	4
Test Name	Calendar
Purpose	This test case tests the functionality of the calendar tab.
Precondition	The user is logged in.
Assumptions	
Steps	The user selects a date on the calendar.
Expected Outcome	Date activity is displayed with selected date.
Actual Outcome	As expected.

Table 7.5: Calendar Test Case

### 7.2.6: Planner Test Case

Test Case ID	5
Test Name	Planner
Purpose	This test case tests the functionality of the planner tab.
Precondition	The user is logged in.
Assumptions	
Steps	The user selects the add button on the planner tab. The user enters in their information. The user submits schedule.
Expected Outcome	Schedule is saved and calendar tab is displayed.
Actual Outcome	As expected.

Table 7.6: Planner Test Case

### 7.2.7: Contacts Test Case

Test Case ID	6
Test Name	Contacts
Purpose	This test case tests the functionality of the contacts tab.
Precondition	The user is logged in.
Assumptions	
Steps	The user selects the "Contacts" tab
Expected Outcome	Contacts are returned from the database and displayed in a list.
Actual Outcome	As expected.

Table 7.7: Contacts Test Case

## 7.3: Evaluation

Testing is a very important part of the software development life cycle. It is very important that good methods of testing are implemented to insure the project has a successful testing phase. This will help the final result to be better. Testing is very important because the application remains in a constant state of change. For this project extensive testing was carried out with test cases. Each function tested worked as expected.

# Chapter 8: Conclusions and Further work

## 8.1: Conclusion

In order to conclude that this project has been successful, the following questions need to be looked at again from the start of the project:

- Determine if this application is unique.
- Determine if this application improves on existing applications in the medical field.
- Determine if there is an actual need for an application like this in medical practices.

Looking at the first question we can confidently say there is no application on the market which provides all of the features that this application does. Looking at the second question we can definitely say that this application has made improvements on existing application by adding in additional features and improving on existing ones. And finally looking at the third question we can happily say that there is a demand for an application like this in medical practices after conducting the survey at a medical practice which received very positive feedback. The main purpose of this project was to explore the possibility of creating an android application that allows users to manage their medical needs in a different and more efficient way. I am happy to say that this project has fulfilled its purpose based on its feedback.

## 8.2: Further work

Although the project has achieved its desired outcome and is fully functional, there are some additional features I would have liked to add in if I had more time, such as:

- A website for the application.
- Allow users to upload images to their profiles.



# **Appendix A: Project Planning**

## **October**

This is the month when I decided on the type of project I wanted to do. I decided to do an android application because android is an area of computing which interests me and is an area I would like to work in. I also wanted to learn more about android and I saw this as good opportunity to do so. I chose to do a medical management application as I had seen other applications in the medical field and I felt that I could improve on these applications.

## **November**

This is the month when I started to learn more about android and how to develop applications with Android. It was a new experience to me as at the time I had just started to use android. During this month I also completed my project proposal.

## **December**

In this month I carried out a substantial amount of research on mobile applications, specifically android applications and applications already available in the healthcare system. My literature review was completed during this month.

## **January**

During this month I had my end of semester one exams so I didn't get to make as much progress on the project as the previous months. Although I did get to make some decent progress on the system design.

## **February**

During this month I was able to get the database up and running and communicating with the application through PHP scripts.

## **March**

This is the month I had two weeks off for Easter. I was able to make strong progress during this time. I was able to get a lot of the features working on the application.

## **April**

By now, I had most of the application functioning. I carried out a survey on the application to get feedback about the application.

# List of References

- [1] Book Doctor Appointment / *Google Play*, Retrieved 4/4/2016, 2016, from <https://play.google.com/store/apps/details?id=com.bda.patientapp>
- [2] Shao Guo-Hong, "Application development research based on android platform," in *Intelligent Computation Technology and Automation (ICICTA)*, 2014 7th International Conference on, 2014, pp. 579-582.
- [3] Developers A. What is Android[J]. 2011
- [4] ZHANG S C. Development and Research of Application Based on Google Android [J][J]. *Computer Knowledge and Technology*, 2009, 28.
- [5] Brahler S. Analysis of the android architecture[J]. Karlsruhe institute for technology, 2010.
- [6] Danyl Bosomworth, *Mobile marketing statistics* 2015
- [7] Nielsen, *Who's Winning the U.S. Smartphone Market*, 2013
- [8] T. Bradshaw, "Apple's app advantage wanes as developers warm to Android," *The Financial Times*, pp. 1, 12/28; 2015/11, 2012.
- [9] A. Stevenson, "Top 10 Android benefits over Apple iPhone," *V3.Co.Uk*, 03/28; 2015/11, 2014.
- [10] M. Rowan and J. Dehlinger, "A Privacy Policy Comparison of Health and Fitness Related Mobile Applications," *Procedia Computer Science*, vol. 37, pp. 348-355, 2014.
- [11] Rowan, M. and Dehlinger, J. Privacy incongruity: an analysis of a survey of mobile end-users. 13th International Conference on Security and Management. 2014.
- [12] Steinberg, J. (2013, December 6). This flashlight android app has been secretly and illegally sharing your personal data with advertisers. Retrieved from: <http://www.forbes.com/sites/josephsteinberg/2013/12/06/this-flashlight-android-app-has-been-secretly-and-illegally-sharing-your-personal-data-with-advertisers/>.
- [13] Slabodkin, G. (2014, May 8). FTC concerned with health data sharing apps. Retrieved June 10, 2014 from: <http://www.healthdatamanagement.com/news/FTC-Concerned-with-Health-Data-Sharing-Apps-48017-1.html>.
- [14] Dubay, W. (2004, August 25). Principles of readability. Retrieved June 12, 2014 from: <http://www.impactinformation.com>.
- [15] Phandroid Forums. (2012, October 31). Confused over app permissions. Retrieved June 11, 2014 from: <http://androidforums.com/android-applications/641501-confused-over-app-permissions.html>.
- [16] Google Play. (2014). Review app permissions. Retrieved June 10, 2014 from: <https://support.google.com/googleplay/answer/60149727>.
- [17] H. J. Seabrook, J. N. Stromer, C. Shevkenek, A. Bharwani, J. de Grood and W. A. Ghali, "Medical applications: a database and characterization of apps in Apple iOS and Android platforms," *BMC Research Notes*, vol. 7, pp. 573, 08/27; 2015/11, 2014.
- [18] Wallace S, Clark M, White J: ?It?s on my iPhone?: attitudes to the use of mobile computing devices in medical education, a mixed-methods study. *BMJ Open* 2012, 2(4): 1-7.

- [19] Mosa ASM, Yoo I, Sheets L: A systematic review of healthcare applications for smartphones. BMC Med Inform Decis Mak 2012, 12(1): 67
- [20] S. Challa, G. Geethakumari and C. S. N. Prasad, "Patient data viewer: An android application for healthcare," in India Conference (INDICON), 2011 Annual IEEE, 2011, pp. 1-4.
- [21] R. B. Alday and R. M. Pagayon, "MediPic: A mobile application for medical prescriptions," in Information, Intelligence, Systems and Applications (IISA), 2013 Fourth International Conference on, 2013, pp. 1-4.
- [22] Jeremy C. 2007. Cause of Death: Sloppy Doctors <http://www.time.com/time/health/article/0,8599,1578074,00.html>
- [23] E. Sugiono, Y. Asnar and I. Liem, "Android security assessment based on reported vulnerability," in Data and Software Engineering (ICODSE), 2014 International Conference on, 2014, pp. 1-6.
- [24] C. Nachenberg, "A Window Into Mobile Device Security," Symantec, 2011.
- [25] Google. (2014, Jun.) Google Developer. [Online]. <https://developer.android.com/about/dashboards/index.html> [Last Accessed on June 2014, 25]
- [26] NIST, SP800-3-Risk Management Guide for Information Technology Systems. Gaithersburg: National Institute Of Standart and Technology, 2002.
- [27] C. P. Pfleeger and S. L. Pfleeger, Analyzing Computer Security: A Threat/Vulnerability/Countermeasure Approach. New jerseys: Prentice Hall, 2012.
- [28] Security Tips | Android Developers, [Last Accessed on 11/29/2015].
- [29] Software Development Life Cycle(SDLC) Spiral Mode | *Simplicity Through Breadth*, Retrieved 4/4/2016, 2016, from <https://faisalsikder.wordpress.com/2009/12/18/software-development-life-cyclesdlc-spiral-model/>