NHS Wales - Injectable Medicines Guide Android application

Final Report for CS39440 Major Project

Author: Aidan Wynne Fewster (awf1@aber.ac.uk)
Supervisor: Dr. Andrew Starr (aos@aber.ac.uk)

May 2014
Version: 1.0 (Release)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G400)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Acknowledgements

I'd like to thank Julie Fewster, Wynne Fewster and Hannah Seabridge for their motivational support. I'm grateful to thank Robin Burfield and Simon Watkins, for providing the API resources and any help I needed throughout the project. I'd also like to thank the developers of Anrdroid, Robospice, JUnit and LaTeX.

Abstract

The Injectable Medicines Guide is an Android application, which was created for the NHS Wales under the Software Alliance Wales Scheme. The Injectable Medicines Guide aims to aid NHS staff in administering injectable medicines to patients, by providing portable access to a dataset of drug monographs.

Drug information is downloaded from a database provided by the NHS, this data is then stored locally on the device, so that it can be used when no Internet connection is available.

The downloaded data is then utilised by the application, allowing the user to search through the entire list of downloaded drugs. Once the user has found the drug they were searching for, they can select it and will displayed with a detailed monograph of the selected drug. The monograph is displayed neatly to the user with helping information where needed.

The Injectable Medicines Guide provides the user with a calculator for many of the drugs inside the database. The calculator can be used to calculate both dose and infusion rate values for a specific patient. When a calculation is performed the result of the calculation and a breakdown of the calculation is displayed to the user.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background, Analysis and Process

This chapter contains evidence of the background work that was completed before the project design was created. During this stage in the process existing works were considered, an analysis of the problem was complied, methodology chosen and a plan was created.

1.1 Project Overview

The NHS [34] owns a database containing monographs [33] for injectable medicines; each monograph contains useful information such as method of administration, preparation of the drug and flushing guidelines. As well as monographs for each medicine the database also contains values needed for calculating dosage and infusion rate for the medicines.

NHS Wales [34] requested Aberystwyth University via the Software Alliance Wales scheme [43] to provided them with a mobile application, to utilise this data to aid their staff in administering injectable medicines. The NHS provided access to the database via multiple XML [49] API URLs.

The aim of this project was to fulfil that request by creating a well designed, functioning and thoroughly tested Android [9] mobile application. The completed application had to query the data to allow users to quickly and efficiently find monographs. Upon finding the wanted monograph the application has to neatly present the monograph to the user. As some medicines also contain data on calculating dosage and infusion rates, the application had to utilise that data and allow the user to enter patient information (weight and needed concentration) to calculate dosage or infusion rate for that patient.

An Internet connection may not be available [19] in all areas of a hospital, to allow the application to be used whenever needed, the application has to be built for offline use, to achieve this a complete copy of the database has be stored on the users device, thus allowing them to utilise the data when no Internet connection is available.

As to improve the maintainability and customisability of the system the NHS also requested that the structure of the data to be outlined within XML [49] files, thus allowing them to create multiple applications for a variety datasets using the same application code.

As the application will be used to administer potentially lethal drugs, testing had to be executed thoroughly. Therefore a major part of this project was testing the finished application, ensuring that only the correct and the most accurate information is displayed to the user.

1.2 Background and Analysis

This section will outline all background research that was executed before starting this project. This will include a list of the available technologies that could have been used and an overview of what technologies already exists that complete similar goals to this project.

1.2.1 Medusa website

The Medusa [20] website is the site that NHS staff currently use to view and print monographs for injectable medicines. The website requests the member of staff to login using their NHS credentials, upon logging in a user is able to select a drug from a drop down list (suggestive search is not available). Once a drug has been selected the user is displayed with the monograph [33] for the selected drug. If calculator information is available, then a button to open the calculator is also displayed.

The Medusa website [20] does not work on the mobile devices that I tested it on (the login functionality fails) and the site is not optimized for the screen size of most mobile devices. As the Medusa website [20] does not work effectively on mobile devices the only method of accessing the monographs [33] in a portable manner is to print the monographs, this not an optimal solution as the printed information may become outdated and the user will not know, also this is not an environmentally friendly method.

Using the Medusa website [20] allowed me to see how a monograph [33] should be displayed to the user. It also allowed me to see the shortfalls and drawbacks of the current system, such as the lack of a suggestive search functionality, which allowed me to change the projects requirements to provide a solution to these issues.

1.2.2 Platforms and frameworks

Due to the time constraints of this project, one of the major decisions for the project was which framework or platform should be used, to allow the application to be compatible for the largest amount of users.

A solution that would have allowed the application to run on the majority of devices would have been to create a web application [50]. A web application [50] is essentially a website that has been optimized for a mobile device. The major issue with a web application was that there is no way to store data on the device persistently. Therefore a network connection would have been needed, as the application has to work in offline mode [19], a web application is not a viable solution.

Phonegap [35] is a mobile development framework that allows developers to write mobile applications using HTML5, JavaScript and CSS3 instead of device specific languages. Phonegap [35] then compiles the application, that is written using web technologies, into a hybrid application [51] for multiple devices. A hybrid application [51] is an application that appears to be a native application to the user, but is not as instead of using the devices UI framework the application uses web views to display information to the user. The framework also gives developers access to the devices local storage and sensors, thus allowing developers to create web applications with similar powers to native applications.

Phonegap [35] would have been an excellent framework to use for this project, as it would have allowed me to create applications to be used on multiple devices, using languages I was very familiar with. The main issue with Phonegap is that it does not allow you to run processes in the background [5]. Therefore the process of downloading and updating the local database would need to run in foreground, which is not an ideal solution as the process takes several minutes and the user is likely to move out of the application while the download is in progress. Another issue with Phonegap [35] is that as the application does not utilize the devices UI frameworks the application may lack the look and feel of a native application, meaning it may be harder for a user to use the application.

Native Android [9] and iOS applications would have been suitable for the project. They both have the ability to download the data over HTTP, parse the XML [49] files, execute tasks in the background and save data for offline use within local databases.

Applications for iOS [16] devices are created using the Xcode IDE and are written in Objective-C, which is an object-orientated language based on C and C++ [38]. I have some experience in C and C++ but no experience in Objective-C or iOS development. I also do not own an iOS device therefore testing would have been primarily executed within an emulator.

Android applications are written in Java [29] which is an object-orientated language using either Eclipse [18] or Android Studio [14] IDE's. I have had a large amount of experiencing writing applications in Java, but had no experience in Android development. I also have two Android devices, which would allow me to test the application on a live device rather than an emulator.

Android currently has the largest percentage of market share, having 78% of the market share in 2013 [10]. Therefore developing the application as a native Android application will allow the application to be used by the most users. Due to this statistic and that I own Android devices, resulted in me choosing to develop the application as a native Android application.

1.2.3 Learning Android development

As I had very little experience in Android [9] development before starting this project during the background work I also had to teach myself Android development.

I began this process by refreshing my Java [29] skills by reading over previous Java projects and writing basic applications. I then followed the tutorials found within the training section of the Android documentation, this helped me setup the Android SDK [13] and begin building applications. I also read the Android design guidelines [12], which helped me to better design the project.

I then built small prototypes for major sections of the final application. This allowed me to spike any parts of the final application I was unsure about whilst continuing to learn Android.

1.2.4 Existing works

From the research executed it was concluded that there is currently no other mobile application that completes all goals of this project, but there are many libraries, frameworks and tools that were used throughout the project.

The Java [29] programming language provided the application with a great amount of useful functionality allowing the applicaton to complete tasks such as downloading data using HTTP requests, parsing XML [49] into usable data and the ability to write the downloaded data onto the devices internal storage.

The Android SDK [13] allows developers to create native Android applications that have the ability to utilise all hardware on a users device. The Android SDK [13] also provides the base UI framework, allowing developers to create applications that an Android user can instinctively use.

Robospice [39] is a library for Android that is released under the Apache licence. Robospice simplifies the process of making asynchronous network requests in the background whilst continuously notifying the UI thread of progress [39] . The Robospice [39] library has been used to allow the complete database download to be executed in the background as an Android service, whilst still updating the user interface showing progress to the user.

Glyphicon [25] is an open source free to use iconography package. This project uses one icon from the iconography set to display an information button.

Another resource that was used whilst completing this project was StackOverflow [46]. StackOverflow is an online community where users post programming related issues and the community help solve these issues [46]. StackOverflow was used when an issues was encountered that I believed would be a common issue or when best practices for a solution were unknown by myself. I never posted any issues of my own, just read other users posts.

Genymotion [22] is an application that creates and runs emulators for a variety of devices running Android. In my experience Genymotion is quicker and less error prone than the standard Android SDK [13] emulator. I used Genymotion as it allowed me to test my application on multiple devices without requiring me to setup each device individually, as I would have had to with the standard emulator.

1.3 Objectives

Within this section I will outline the objectives of the project and state how background work helped reach these objectives, I will then state the final list of functional requirements used for the project.

1.3.1 Original objectives

Build a native Android application using the Android SDK Building a native application for each of the major platforms would have been the ideal solution for this project, but due to the time constraints for this project it would only have been possible to create one application. After completing the initial background work I decided that I would be developing a native Android [9] application, as this approach would allow the application to be used by the most users [10].

Use the user interface framework provided by the Android SDK Using the user interface framework provide by Android allowed me to create an application that a user already familiar with the Android OS would be able to use with ease. This also allowed me develop the interface quickly instead of having to design each element before hand.

Easy to use user experience The NHS staff that will be using the application might not be technically minded therefore the application must be easy to use for people with little technical knowledge. I achieved this by following the design guidelines within the Android documentation, which I read during the background research for the project.

Download the database so the application can be used without an active Internet connection As the radio waves used to transmit data over Wi-Fi or the mobile data network can effect medical equipment [19] it is vital the application runs perfectly in airplane mode, through background research I found that the best way to achieve this was to store all data needed for the application on the devices local storage. Using this method the user can download the database when they have an Internet connection and then continue to use the data when they're without an Internet connection.

Thoroughly test the application throughout all stages As the application is used to administer drugs to real patients it is vital that all aspects of the application are thoroughly tested. Therefore a test-driven development approach was taken towards classes that output life critical information [28]. I believe once the application is given to the NHS they will also execute their own tests, but providing my test data and documenting these test should greatly improve their confidence in the application.

Implement the application using only publicly licenced libraries and resources As the NHS will use the application all libraries and resources used must not be for personal use only and therefore should be licenced under publicly free licences such as the Apache licence.

1.3.2 Functional requirements

After building the list of original objectives I contacted the representative of the NHS for this project and through this contact, the list of API URLs provided by them and the initial mock-ups sent through email I was able to compile the following list of sensible functional requirements.

Table 1.1: Table of functional requirements

FR 1	**Authentication**
FR 1.1	User must be able to authenticate themselves using their credentials
FR 1.2	User must be notified if the password they enter is incorrect
FR 1.3	User will be notified if the authentication failed due to connection issues
FR 1.4	User must be able to logout of the system, removing all data
FR 2	**Database synchronisation**
FR 2.1	After login or when the user presses update the application must truncate all database tables and begin downloading new data

FR 2.2	Download complete list of drug indexes from database
FR 2.3	Download complete list of drugs and drug information's
FR 2.3	Download all information needed for calculating doses and infusion rates.
FR 2.4	Download must still run when the application is in the background
FR 3	**Menu options**
FR 3.1	Upon pressing the Menu button on the device the user will be presented with a list of available options, which execute tasks (Logout, exit, search)
FR 4	**Main screen**
FR 4.1	Upon successful data download the user will be displayed with a screen where they can navigate to other parts of the application
FR 4.2	User will be see when an update was last performed, and perform an update from this screen.
FR 5	**Browse drugs**
FR 5.1	This screen will allow the user to view a list of all drugs
FR 5.2	There will be an input box on this screen, when the user enters text into the input box the results in the list will be filtered to only show results related to the input
FR 5.3	The user will be able to click a drug in the list to open a new screen displaying the needed information
FR 6	**View drug**
FR 6.1	When a drug has been selected the drug and all it's information will be displayed in an easy to read format
FR 6.2	Where drug information headers contain help information, a help icon will be displayed next to the header.
FR 6.3	When heading help icon is clicked the helping information will be displayed
FR 6.4	If the drug had calculator information, then a button to open the calculator should be shown
FR 7	**Browse drugs with calculators**
FR 7.1	A view similar to the browse drugs view will allow the browsing of drugs that contain calculator information.
FR 8	**Calculate dose and infusion rate**
FR 8.1	The user will be able to select calculation type
FR 8.2	User will be able to enter information required for the calculation
FR 8.3	When the calculate button has been clicked the input will be thoroughly validated
FR 8.4	After validation the result of the calculation will be displayed to the user
FR 8.5	The equation and values used to calculate the answer will be neatly displayed to the user
FR 9	**XML customisability for developers**
FR 9.1	All text within the application must be changeable through XML files.

FR 9.2	The structure of the XML API's provided must be outline within XML files, allowing easy customisation for different API's

1.4 Compromises within the functional requirements

Creating a perfect system that met all the functional requirements that were wanted was not possible due to time constraints or API limitations, within this section I will be outlining the functional requirements I would have liked to have added but were not possible.

As mentioned earlier, creating a native application for each major platform would have been ideal for this project, but due to time constraints and the lack of a usable hybrid solution this was not possible.

Having the database synchronise with the live database, instead of deleting the entire database and downloading a new copy would have been a more efficient method of updating the database, as updates would be performed much quicker. Unfortunately synchronising the databases was not possible as the API provided only allowed for downloading of the complete dataset and not partial downloads.

Sending push notifications to the device when a change to the database occurred would have been an excellent way to alert the user that they needed to update their data, but due to lack of access to the server hosting the database this was not possible.

Allowing the user the ability to reset their login credentials should they forget them was also not possible due to API limitations.

1.5 Development process

Within this section I will be describing the methodology that I used whilst completing this project, explaining my method of planning the system and describing the prototypes that I created to help break down the project.

1.6 Methodology

Initially an eXtreme programming [2] methodology adapted for solo programming was the chosen methodology for this project. Using an eXtreme programming approach would have allowed me to create good working software faster than most other methodologies whilst preventing the project from being hacked together.

Within the initial methodology it was planned that I would engage in meetings with the NHS representative to allow us to collaboratively create a set of stories and a detailed releases schedule for the project.

After creating stories for the project the methodology stated to carry out spike work by completing prototypes for predicted difficult stories of the project thus gaining a better understanding of the complexity of the stories. These prototypes could then later be used to demonstrate work completed during the mid-project demonstration.

Once the complexity of all stories had been estimates and spike work had been carried out it was intended that a meeting with the NHS would be organised so that stories could be ranked in order of importance, then stories with greater importance could be implemented first, thus allowing the application to made useful to the NHS sooner.

As I started working on the project I soon released that an eXtreme programming approach to completing this project would likely lead to failure, this was mainly due to the fact that regular contact with the NHS representative was not possible. I also began to believe that using eXtreme programming [2] might not be an acceptable methodology for a medical system because maintaining a system created by another developer that is not well documented would be hard and error prone.

After realising that eXtreme programming [2] was not the best methodology for this project I decided to move the project to use the waterfall model [3], whilst still using features from eXtreme programing [2] that I believed would enhance the project. Using the waterfall model [3] would allow the project to be completed with as little contact with the NHS as possible whilst enforcing me to create a list a detailed documentation to be delivered along with the project

Using the waterfall the model meant that a detailed analysis of the project would be needed, this was completed whilst carrying out background and analysis work as seen in the previous section and was compiled into the outline specification document [3].

The next stage in the waterfall model was to compile a list of requirements; the list of requirements along with a use case diagram and description was used to create the requirements specification.

Using the functional requirements, prototypes were then created for any requirements that the complexity was unknown for. This was a practice that I had taken from the eXtreme programming methodology because I believe that creating a prototype for a task that seems challenging helps to simplify that task.

Once the prototypes had been built for the major parts of the system the next step was to design the application. For this I followed the waterfall model's method of design and created a design specification containing user interface design, UI mock-ups and class structure and design using UML diagrams with appropriate descriptions [3].

After the design process for the application was complete implementation began. Implementation followed the structure laid out in the design specification and used the prototypes created to build a working and well-engineered Android application.

Test-driven development was used for parts of the system that outputted important information. Test-driven development is a practice that I integrated into my methodology from eXtreme programming. Using test-driven development for critical parts of the system ensured that those parts of the system were well tested.

During the implementation stage the continuous integration practice from eXtreme programming was used so that the applications was always in a state where it could be compiled and ran on a device. This motivated me to continue developing the application as I could see the application gradually becoming more useful.

Another practice that was taken from eXtreme programming [2] was the use of coding standards. Throughout the implementation and testing process all code followed the Android developer coding standards and JavaDocs comments were created.

The final step in the waterfall model [3] is to verify and test the implemented application. During this stage the parts of the applications that remained untested were tested and further testing was carried out on the already tested parts. The application was then sent to the NHS representative for acceptance testing.

1.7 Planning

Planning is vital to the success of a project and was inherently brought to the project through the waterfall method [3]. Both the analysis and design stages of the waterfall model are planning stages therefore completing this stages ensured the project was well planned.

After the analysis stage of the project and a full list of dates for the project had been published, I produced a Gantt chart plan for the project time span. I set out the Gantt chart plan to fit with the academic calendar for the module, which ensured I had a large amount of work complete before the mid-project demonstration. I also planned the Gantt chart so that to allow for 3 weeks of spare time at the end of the project. This extra time was reserved in case I encountered any unseen problems during any stages of the project, meaning I wouldn't have to worry about time constraints should this have happened.

Throughout the development process the projects progress was compared to the Gantt chart plan, which allowed me to keep track of how far ahead or behind the project was with the timeline.

During the time when I had intended to begin implementing the application the representative for the NHS was unavailable and I had to wait until they were back in the office until I could further progress the project. Having already planned for an unexpected event happening such as this greatly reduced the impact on the project.

1.7.1 Prototypes

Before starting the design stage of the waterfall model [3] I wanted to improve my Android skills whilst still enhancing the project. To do this I decided to build multiple prototypes that complete small tasks for the final implementation. This section will describe each prototype made and what they achieved.

NHS Prototype Package A Java [29] package containing common models that would be used between prototypes was created, this ensured that the prototypes would work together flawlessly once implemented into the final application whilst also making the code written DRY.

View Drug Prototype The main purpose of this prototype was to display a drug to the user. All information for the example drug was hard coded into the prototype. The prototype displayed the example drug and all its information onto the screen dynamically. This prototype taught me how to use the Android layout inflator, which was used to display the drug information's. This prototype also taught me how to implement a view that scrolls when the view size exceeds the screen size.

Calculator prototype The purpose of the calculator prototype was to create an application that would accept user entered information and provide the user with an answer to a calculation. At this stage in the project I had not been provided with any information on how the NHS performed their calculations, so the equations used were made up. Although this prototype was never used in the final application, completing this prototype taught me how to accept and validate user input.

Download data index prototype The purpose of this prototype was to download the index data from the drug indexes XML [49] API and then return an array of drugs created from the index. The application begins by asking the user to enter their login credentials and then upon successful authentication the application begins the download of the indexes. The login section of this activity was used in the final application but the actually downloading functionality was not used. Whilst completing the prototype I learn how to receive data over HTTP and how to parse XML within Android. I also discovered several bugs in the XML API provided by the NHS whilst building this prototype, as this was very early in the development I was able to notify the NHS representative to have these bugs fixed.

1.8 Version control and backup

Due to the size of this project, it was vital that a version control system was used and due to the academic value of the work completed its must also be securely backed up. Although there are many backup and version control system available, I only considered two (SVN and git).

SVN is a centralised version control system where as git [23] is a distributed version control system, but as a single developer will develop this project the difference is negligible. One advantage of SVN is that the University can provide students with a free SVN repository.

I have greater experience in using git as I have used git for the majority or my academic and freelance projects. I also have a free Github students account and therefore decided to use git as the version control and to use Github [24] as a backup for the repository. Should the developers of the NHS want to see the process I used to create the application I can share the repository along with all previous versions to them, via Github.

After every considerable change made within the repository the change was committed and pushed to Github. I decided to keep all contents of this project, including all documentation within the repository. This ensured that all data was securely backed up.

Chapter 2

Design

Design is the second step within the waterfall model [3]. This chapter will describe and provide reasons for the architectural design decision made throughout this project. This chapter will also include the steps that were taken to design the user interface.

(Appendix. Design Specification) *For a more detailed description of the design for this project, please see the included appendix entitled Design Specification.*

2.1 Overall Architecture

I intended to design the system in a modular format thus the system contains four main packages with each package containing a tests package. Keeping code modulated enhances the reusability of the code and also makes the code easier to maintain. In this section I will be listing each package and explaining their purpose within the application.

Activities package Activities are Android's views [9]. An activity provides a method of displaying information on screen for the user. The activities package contains the implemented classes for the activities.

Models package The models package contains the model classes for each of the database tables. A model represents an item created from the database, for example a drug or a piece of information about a drug. Keeping all models within one package allows the models to be reused in other applications.

Data download package The data download package contains the Robospice [39] requests classes, which are responsible for downloading the individual API URLs and then parsing the data obtained. Each class within this package downloads a specific API XML file [49]. Having all the data download classes in one package makes it easy to add new classes to download additional data in the future.

Database package The database package contains the class that is responsible for the interacting with the SQLite database [44]. This class is responsible for fetching, creating and deleting data from the database. Having this class within its own package will allow the database and database code to be reused within other applications. If the code within the database helper class becomes to large using a package will also allow the code to be broken into smaller classes.

Tests packages Each package contains a package called tests. Within each test package is classes testing each class of the original package through both JUnit [31] tests and Android instrument tests.

XML edibility As the application will be given to a team of developers who may not have extensive experience in Android development it was essential that I make the basics of the application editable by an inexperienced developer, to achieve this all string values have been set in one XML file (strings.xml) [47]. Having done this, any text within the application can be changed by editing this XML file [49]. As this application is produced for NHS Wales [34] they may in the future want to translate the application to Welsh, adding Welsh language support would be relatively easy due to this design choice.

To further improve the customisability of the application all API URL's and the structure of the XML files is set within a single XML file (data_download.xml) [47]. This makes the task of changing API URL's and the format of the data simple.

2.2 Data storage

One of the important decisions when designing the application was deciding how the data for the application would be stored, within this section I will discuss the possible methods of storing the data and why I chose the method I chose.

2.2.1 Store the data as XML files

Downloading the XML from the XML API URLs and storing the XML file [49] on the device was one of the methods I considered. This method would allow for a quicker download task, as the task would not have to parse the XML. This method would also ensure the data was exactly as it was downloaded.

This is not a great method of storing the data on the device, as the device would have to parse all the XML files every time the data was needed by the application. This method would also be slow at searching and ordering the data, as it provides no indexing functionality.

2.2.2 Store the data as a serialised object

For this method the application would parse the XML file [49] into Java [29] objects and then save the object onto the device through serialisation. This method would be quicker at runtime than storing the as XML files as the data will already be parsed when opening the data.

There are several issues with this method, the first issue being that this method does not allow indexing of the results and therefore searching the data will be slow. Another issue with this method is the entire dataset will need to be loaded into the devices memory whenever using the data, which could cause some devices to run out of memory.

2.2.3 Store the data within an SQLite database

The Android SDK [13] comes with a set of libraries for creating and interacting with SQLite databases [44]. SQLite database can be indexed and are therefore quicker than the above-mentioned methods for searching the data [44].

I decided to use an SQLite database [44] to store the data for the application. The data is parsed from XML [49] into usable objects and then stored in the SQLite database. Using an SQLite database allow the application to only load needed data into memory when needed, by using the WHERE operator when searching for data. Using an SQLite database also helped keep the data in an organised structure, which would allow the data to be used by other applications if needed.

The main issue with using SQLite database [44] is that the stored data may be larger than if the data was stored as a serialised object, but if the size of the database became an issue the database could be compressed.

2.2.4 Database design

As it had been decided that a database would be used to store the data, the next step was to design the database structure. The database tables have been normalised into third normal form. SQLite foreign key constraints were only introduced into the Android SDK [13] at API version 16, as I wanted the application to have the largest reach possible I decided not to use foreign key constraints and therefore monitor foreign key relations manually.

The first table to be designed, was the drugs table. The purpose of this table is to contain all the drugs that are downloaded. Within the original designs I had planned to place all drug information within the drugs table, but as the drug information's changed between drugs I decided it would be better to create a separate table for drug information's.

Table 2.1: Table showing drugs database table structure

Table name	**Data type**	**Allow null**	**Purpose**
id	Integer	No	Holds the drugs ID that is set from the downloaded XML. This is the primary key.
name	String	No	Contains the drugs name.

The next table to be designed, was drug information's table. The purpose of this table was to hold a single piece of information about a drug, such as the method of administration or the flushing guidelines. Some drug information's also contain a header helper. Header helpers are extra help information explaining the header. These are displayed to the user upon clicking the header or information button next to the header.

Table 2.2: Table showing drugInformations database table structure

Table name	**Data type**	**Allow null**	**Purpose**
id	Integer	No	Holds the drug information id, this is an auto incrementing value. This is the primary key
drug_id*	Integer	No	The id of the drug this information belongs to. This is a foreign key to the drugs table id.
header_text	String	No	The header for the piece of information
header_helper	String	Yes	The helper information for the drug information if it exists
section_text	String	No	The main body of information for the drug information.

The next table that was designed, was the drug indexes table. This table contains the drug indexes. A drug index is an extra name for a drug. A drug may have more than one index. The drug indexes are what the user will be shown in the list of searchable drugs.

Table 2.3: Table showing drugIndexes database table structure

Table name	**Data type**	**Allow null**	**Purpose**
id	Integer	No	Holds the drug information id, this is an auto incrementing value. This is the primary key
drug_id*	Integer	No	The id of the drug this information belongs to. This is a foreign key to the drugs table id.
name	String	No	The name for the index

The final table to be designed, was the table containing the drug calculation information's. This table contains all the information needed to perform the infusion rate or dosage calculations. As SQLite [44] does not support boolean types [45] integer values (0 or 1) were used to represnt booleans. Integers representing booleans were used for if the patient weight required and if time is required columns.

Table 2.4: Table showing drugCalculations database table structure

Table name	**Data type**	**Allow null**	**Purpose**
id	Integer	No	Holds the calculators id, this is an auto incrementing value. This is the primary key
drug_id*	Integer	No	The id of the drug this information belongs to. This is a foreign key to the drugs table id.
infusion_rate_label	String	No	The label used for the infusion rate calculator
Infusion_rate_units	String	No	The units of the calculated infusion rate.

dose_units	String	No	The units of the calculated dose
patient_weight_req	Integer	No	If the patients weight is required for the calculation (1 represents true, 0 false)
time_req	Integer	No	If the time is required for the calculation (1 represents true, 0 false)
factor	Integer	No	The factor used within the calculation

2.3 Downloading of data

The next part of the design process was to decide how the data for the database would be downloaded. The download process had to be able to run in the background on the device as the download can take several minutes, therefore a user is likely to minimise the application. In this section I will discuss the methods of downloading the data that I considered.

2.3.1 Download the data using an AsyncTask

It would have been possible to download the data using an AsyncTask [17] as AsyncTasks create a new thread meaning they can be used for HTTP requests. Within the Android documentation it states that AsyncTasks should not be used for long running tasks (more than a few seconds) [17], due to this reason I decided not to use AsyncTasks for downloading the data. Another reason against using AsyncTasks is that they are attached to the activity that created them, so if the user minimises the application the task will also be killed, thus not allowing background downloading.

2.3.2 Download the data using a service

The Android documentation suggests using a service [41] for any long running task [17]. A service runs within its own thread therefore allowing the download to continue working whilst in the background. A service will also place an item in the devices notification panel letting the user know that the service is still running. Using a service to download the data would have worked and been a good solution if implemented correctly.

A service is only a basic class for running a process in the background, it provides no implementation for alerting the view of its progress, nor does it contain any error handling should the tasks fail [41]. The reason I chose not to download the data using a basic service is that I would need to implement a large amount additional code to handle all areas of failure and to notify the view of the downloads progress.

2.3.3 Download the data using a Robospice service

Robospice [39] is an open source library released under the Apache licence [15] that simplifies the implementation of long running tasks. Robospice runs on top of the Android service providing implementation for frequently used features needed when implementing services. Robospice makes notifying the view of progress simple and also allows for multi-threading meaning that simultaneous downloads can occur. Another useful feature that Robospice contains is its error handling features, it retries a task a set amount of times [39] before finally alerting the main thread of its failure.

I decided that using a Robospice service to handle the downloading of data would be the best solution. Using a Robospice service meant I would be able to focus on the body of the task instead of worrying about implementing the various features needed for the service [39]. Robospice does allow for caching, but as the data will only be downloaded once every few months I decided using an un-cached service would be acceptable.

2.4 Design pattern used

When planning the application it was planned to create the application using a Model-View-Controller (MVC) [1] design pattern. Once I began developing within Android it became apparent that a true MVC approach was not possible within Android. Although a true MVC approach was not possible I attempt to follow the MVC pattern throughout my design, thus creating a package for models, a package for activities (views) and separate packages for everything else (controllers). I believe that by modulating the code, the maintainability and reusability of the code has been increased.

2.5 Database interactions class

A single database interaction class was designed. The purpose of this class was to handle all interactions with the database, thus keeping all SQL code into a single class. This class contains the method for creating and retrieving data, whilst also containing the methods for creating and updating the database.

I wanted to be able to test all methods within the database class without altering the original database, to achieve this a separate constructor was creating which prepends a suffix to the database files name, thus making the test database separate from the original one.

Currently the amount of methods within this class isn't very large, but if the application grows it would be a good idea to split this class into separate classes that handle different section of the database, for example a separate class for each table in the database.

2.6 Download data singleton

It had been decided that the downloading of data would be executed within a Robospice service [39]. Robospice services like Android services run within there own thread [39] and are therefore are not attached to a specific activity. Although Robospice services allow for the passing of progress from the service to the activity, they do not natively support the passing of data.

To pass data between the download services and the download activity I designed a singleton class. A singleton class is a class that can only have one object of itself instantiated [4]. A method is created within the class for the purpose of retrieving the single instantiated object. Using a singleton for the download progress meant that both the service and the activity would be using the same object, thus allowing data to be passed between them.

2.7 Calculation class

The calculation class is the class responsible for handling the dosage and infusion rate calculations. I intended to create a powerful class that could handle both the validation and calculation of the calculations.

A method called validate was designed, this method checks that all data used for the calculation is present and correct. The method then returns an integer for the result of the validation. This integer can either represent an error (which the user must change), a warning (which the user has the option to change or continue) or success.

Once the validate method has been called, if the result is a success or a warning and the user opted to continue the calculation will be performed.

Keeping the validation and calculation within the same class means that the calculation code be reused within other projects. Using this approach also allowed for more rigorous testing as the validation and calculations could be tested together whilst Unit testing [31].

2.8 Preferences class

Android provides developers with a class for storing values on the device [9]. This class is called SharedPreferences. A shared preference [42] is a key-value pair accessed by providing the key as a string. SharedPreferences are useful for storing single pieces of information on the device without creating a separate table in the database for them [42]. Android also provides security on top of SharedPreferences, preventing other applications for accessing your applications SharedPreferences.

The preferences class was used as a wrapper class for inserting, updating and deleting SharedPreferences. The preferences class is used within the application to store the users credentials, the date of the last database update and if the database was successfully downloaded.

Although Android provides some security for SharedPreferences [42], they cannot stop a malicious user from accessing the data should they have physical control of the device. Therefore it is important that secure information is not stored within the preferences without being encrypted.

2.9 Authentication class

This class was designed to provide authentication with the NHS server. This class contains methods for validating a users username and password and retrieving the user credentials so that they can be used when updating the database. The authentication class also provides a method for checking whether a user is logged in, this is used when the application is first opened and decides if the login or main view should be opened.

2.10 Splash activity

When the application first opens the application has to decide whether to open the login screen, the main screen or the update screen. Android does not allow any logic to be processed before the first activity is displayed. This caused an issue, as there was no way of opening the correct activity without first opening an activity.

One solution to this problem would be put the logic within the main activity and if the user needed to login or download the database they could be redirected. This would not be an ideal solution as for a brief amount of time the main activity would be opened, causing a flicker.

I decided that designing a splash activity would be the best way to solve this problem. The splash activity briefly displays the NHS [34] logo on a neatly presented background whilst the application decides where to redirect the user, the user is then redirected to the correct screen, without any flickering.

2.11 Common activities

Android provides the user with menu options when the user presses the menu button on the device [9]. This menu contains items such as update, logout and exit. The menu buttons in most applications vary between activities, but within this project the menu items are the same throughout the application, the only time they change is when the user is logged out. It is possible to create the menu item's manually within each activity but I wanted to design the application so that the code is DRY. To achieve this I created a class that inherited Androids Activity class and implemented the Exit and About buttons for that class, this class could then be inherited within activities that needed to display the Home and About buttons. I then created an another class which inherited from that class adding buttons that can be shown once the user is logged in (Logout, Home, Browse, Calculator, Update).

Using this approach makes it easy to add items to the menu of all activities. If an activity needs a custom option adding to the menu this can be done easily within the activity.

2.12 User interface design

I planned to design an application that was both aesthetically pleasing and easy for the novice user to use. Within this section I will discuss how the user interface design decision were made and then display the user interface mock-ups that were created.

2.12.1 Logo design

An easily recognisable icon allows a user to find an application quickly without have to read the name of the application. Android uses the applications logo icon throughout the application [9]. I wanted to create a logo that looked similar to other applications whilst still being unique enough to stand out. I searched the FindIcons website [27] to find an icon that represents medication that was released under a commercially free licence. I then opened the icon in Photoshop, applied a circular background and added effects to the background and icon. The icon was then saved as a variety of sizes to be used on screens of varying densities.

2.12.2 Android developer design guidelines

During the background and analysis stage of the project I read the majority of the design guidelines [12] laid out by Android. These guidelines taught me how to create applications that would scale for a multitude of devices whilst only implementing the design once.

The guidelines [12] taught me to use density-independent pixels (dp), for expressing layout and positioning of view object that need to scale as the density increases. If you were to layout an object using normal pixels, the object would use the same amount of pixel no matter how large the screen was, thus if viewing on a large screen with a large amount of pixels the object would appear small. Using density-independent pixels means the object will automatically scale as the density of the screen does.

I also learnt to use scale-independent pixels for fonts and text [12]. Scale-independent pixels act just like density-independent pixels but they also scale for the users font setting, thus allowing the font to be larger if the user sets a large font within their device settings. Using scale-independent pixels increasing the accessibility of the application. The guidelines also taught me about layout parameters such as match parent and wrap content. Match parent makes the layout parameter an object match its parents width or height [12]. Using match parent is useful when you want a button to fill then the screen. The wrap content layout parameter allows the size of the views width or height to expand depending on the size of its contents This is useful when displaying information to the user which may be more than one line.

Using these learnt Android design techniques allowed me to design a user interface that is fluid and expands well over a multitude of device size, of varying densities. They also allow the views to be rotated depending on the devices orientation automatically.

2.12.3 Designing the interface to be easy to use

In order to ensure that the user interface was easy to use, it was decided that the standard Android user interface library would be used. Using the standard user interface library [12] ensured that a user familiar with the Android operating system would instinctively be able to use the application. To further improve the ease of use of the application I designed a main screen, which opens when the user first launches the application. The main screen describes the main functionality of the application and provides buttons that link to these functions.

2.12.4 Making the application aesthetically pleasing

To make the application as aesthetically pleasing as possible I adapted a simplistic design using a similar colour scheme to other NHS branded services. Simplistic designs have been becoming increasingly popular on the web, as they allow the user to focus on content rather than be distracted by fancy graphics. As the information contained within the application is the most important aspect of the application I decide that a simplistic design would be the best-suited design.

After creating the simplistic design I wanted to keep the user interacted by preventing the application content from being dull. To achieve this I changed the colour scheme of the applications action bar to match the standard NHS [34] blue. This brightened up the application without distracting the user from the main content.

2.12.5 Searching for drugs

When the user navigates to the browse drugs page they are presented with a list containing the complete list of drug indexes. The user can scroll through this list to select a drug that they would like to view. As the list contains over 3000 drugs finding a specific drug manually would take a long amount of time. To solve this issue I designed a search box that would be displayed above the list of drugs, when the user enters text into the search box the list of drugs is filtered to only display drugs that match the searched text. This greatly improves the speed at which a user finds a wanted drug.

2.12.6 Designing the view drugs page

The main challenge when designing the user interface was how to best display the drug monographs [33] to the user. The information contained within drug monographs can be quite large and therefore challenging to fit onto a mobile device's screen. Another challenge added to the designing of the interface was the header helpers. Header helpers are extra information on a header explaining its contents. The view drug page had to have a method of showing that a header has help information and must allow the user to access this information. There were several different methods considered for displaying the information, within this section I will describe each solution that was considered and conclude with the chosen design.

The initial idea and the one provided by the NHS within their mock-ups was to have a view drug page that contained only the drug name and clickable headers, were the clickable headers lead the user to a new page containing the information within that header. This approach would've allowed the data to be displayed to the user and would've fit nicely on a mobile screen. Although this approach would've been suitable for displaying the information, I believe it would not have been a practical solution, as if the user needed information from multiple headers they'd have to navigate into each header individually.

The next design that was considered was similar to the above design, but instead of opening the information within a new page, the information would expand down from the header once the user clicked the header. This design would allow the user to view all the information that they needed at once and it would fit appropriately on a mobile screen. One issue with this approach was that I could not think of a method for showing that the header contains helping information.

The final chosen design was to display the entire drug information in one scrollable screen, using left indents to separate headers and content. This would allow the user to view all the needed within one screen. This design also meant that I could use an information icon next to headers with helper information to show that extra information is available. The on click event of the icon or header text can then be used to also open the helper information.

When the user clicks to view the helper information, an alert dialog box is shown containing the information. Alert dialogs open on top of the current screen and contain a close button to close the dialog [12]. Using a dialog allows the information to be displayed to the user, without them navigating away from the view drug page.

2.12.7 Designing the calculator page

The calculator page is the screen that is used to calculate dose and infusion rates. It is extremely important that the calculated values are correct and therefore the calculator page must be easy for the user to use thus decreasing user errors.

I wanted to keep the calculator page simple, hence why the calculator and view drug page were separate pages. To ensure that the user knew which drug they were currently calculating for, I designed the page so that the drug name was shown in the action bar and within a bold header above the calculator.

The next step in designing the user interface to be easy to use, was to ensure that only inputs that are needed for the calculation are shown. When the user changes between dose and infusion rate calculations, the inputs that are no longer needed are hidden.

To ensure that the user entered the correct values, above each input field is a label describing the input and the units (E.g. Patient weight, kg). This reassures the users that they're using the correct field and that they're using the correct units. Each input also restricts the users keyboard to a decimal keyboard, ensuring the user can only enter numerical values.

To further improve the reliability of the calculator page warnings and errors are shown to the user. Errors are shown when the data enter is not valid, for example a patients weight being less than 0. Warnings are displayed to the user when the value entered seems incorrect, giving the user the option to change the value or to continue with the inputted value.

2.12.8 Showing the calculation

Once the user has entered the values for the calculation and clicked the calculate button they will be displayed with the calculated answer. I decided to use an alert dialog [8] to display the calculated answer. This allows the user to view the answer without leaving the calculator page.

To ensure that the calculation has been calculated correctly I decided to show the original equation used to calculate the dosage, then show the equation with the users entered values and then finally display the answer. This allows the NHS staff member to ensure the application is using the correct equation and also allows them to perform the calculation manually themselves. I believe this approach will improve the NHS's confidence within the application.

2.13 User interface mock-ups

After deciding the features of the design I began creating user interface mock-ups. User interface mock-ups allow you to quickly test a variety of designs without having to complete a detailed design. Mock-ups were created for the application to test designs for each view within the application. Within this section I will list all the user interface mock-ups that were created. The mock-ups within this section were created using LucidCharts [32].

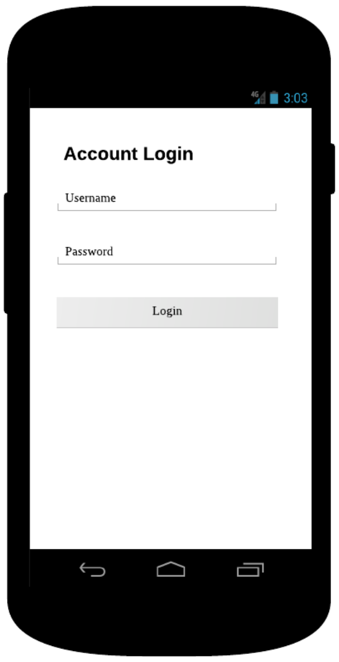


Figure 2.1: Login activity mockup

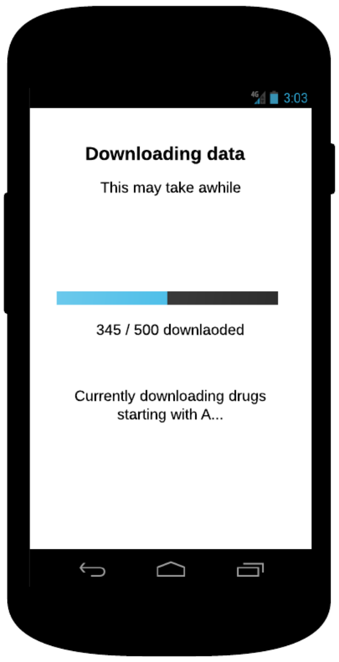


Figure 2.2: Download activity mockup

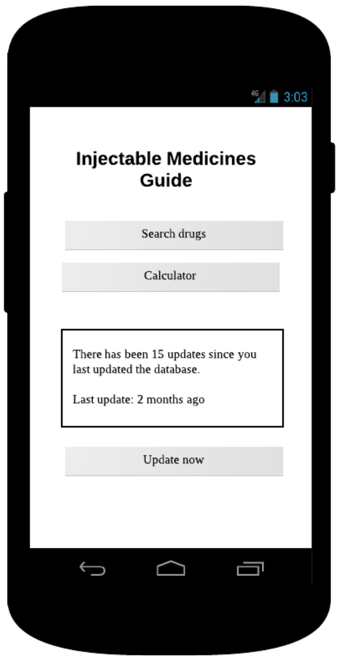

3:03
Injectable Medicines Guide
Search drugs
Calculator
There has been 15 updates since you last updated the database.
Last update: 2 months ago
Update now

Figure 2.3: Main activity mockup

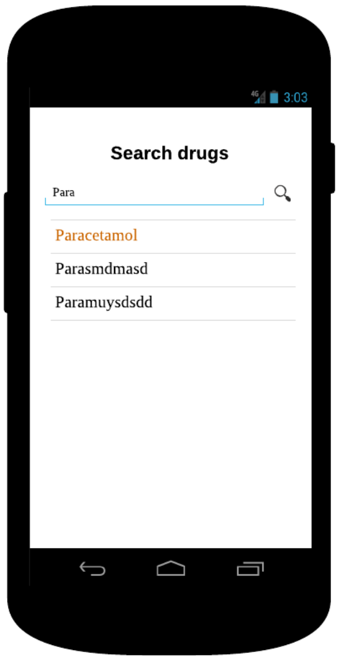
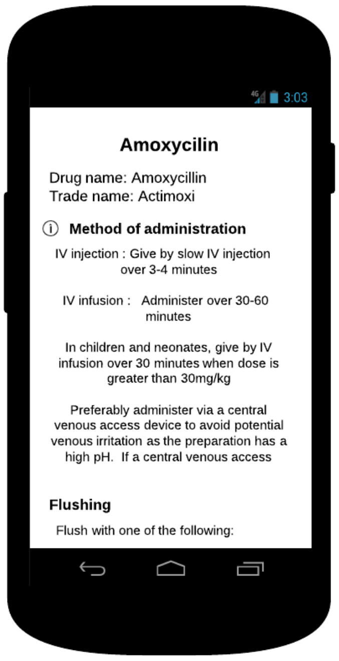


Figure 2.4: Browse drugs activity mockup



3:03

Amoxycilin

Drug name: Amoxycillin
Trade name: Actimoxi

Method of administration

IV injection : Give by slow IV injection over 3-4 minutes

IV infusion : Administer over 30-60 minutes

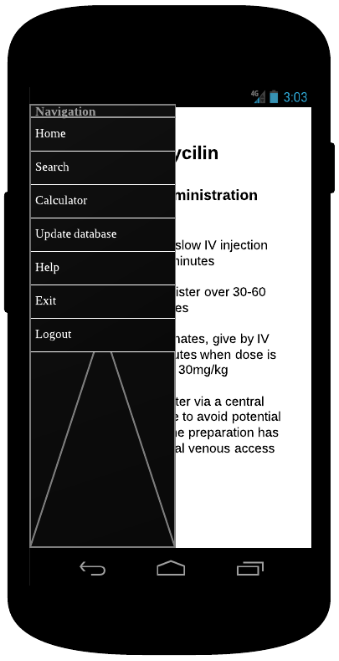
In children and neonates, give by IV infusion over 30 minutes when dose is greater than 30mg/kg

Preferably administer via a central venous access device to avoid potential venous irritation as the preparation has a high pH. If a central venous access

Flushing

Flush with one of the following:

Figure 2.5: View drug activity mockup

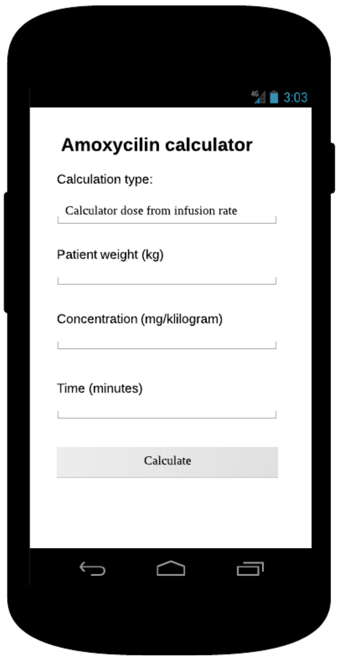


3:03

Navigation
Home
Search
Calculator
Update database
Help
Exit
Logout

ycilin
ministration
slow IV injection
inutes
ister over 30-60
es
nates, give by IV
utes when dose is
30mg/kg
ter via a central
to avoid potential
e preparation has
l venous access

Figure 2.6: Menu mockup



3:03

Amoxycilin calculator

Calculation type:
Calculator dose from infusion rate

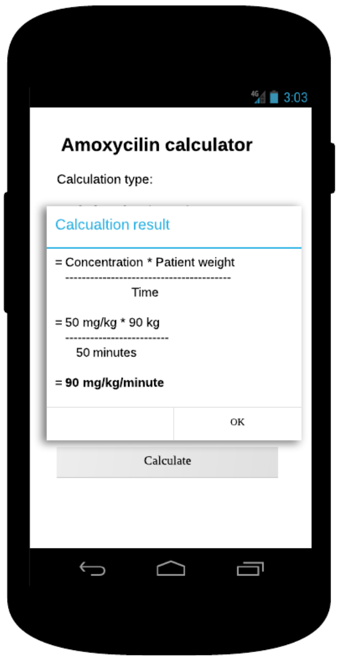
Patient weight (kg)

Concentration (mg/kilogram)

Time (minutes)

Calculate

Figure 2.7: Calculator activity mockup



3:03

Amoxycilin calculator

Calculation type:

Calcualtion result

= Concentration * Patient weight

Time

= 50 mg/kg * 90 kg
----
50 minutes

= **90 mg/kg/minute**

OK

Calculate

Figure 2.8: Calculator display mockup

Login screen This is the first screen the user sees when opening the application for the first time. Once they login their details are saved so they should only have to login once.

Download screen This page is shown when the user is downloading the database for the first time, or when they are updating the database. The tasks within this screen can take several minutes so a progress bar helps to show to the user that the task is still running and hasn't crashed.

Main screen Shown once the user has successfully downloaded the database. This is the first screen a reoccurring user is sent to. This page allows the user to navigate to other screens.

Browse drugs / calculators screen These two screens are very similar; they both display a list of drugs and allow you to filter the list by entering part of the drugs name, inside the search box. The browse drugs page contains a list of all drugs and the browser calculators screen shows all drugs that have calculators.

View drug screen This screen is displayed when the user selects a drug from the browse drugs page. It contains the complete monograph for the drug, some headers contain an icon next to the header, if the user clicks a head with this icon, and helper information for that header is displayed within a dialog.

Calculator screen This is the screen where the user enters patient information and is displayed with the either the dose or infusion rate for the drug. When the result of the calculation is displayed, the equation used to perform the calculation is also displayed and described.

Options menu The options menu is displayed when the user presses the menu button on their device. This menu is designed to allow the user to quickly navigate to useful parts of the system from any screen of the system.

Chapter 3

Implementation

This chapter will include details on the implementation stage of the project. It will discuss the major sections of implementation, the tools used to carry out the implementation and any problems that were encountered whilst implementing the application.

3.1 Overview

The implementation stage of the project went very well, a few errors did arise but these were only ever minor errors and were quickly resolved. Using an IDE helped keep syntax errors to a minimum, which helped decrease the development time. The initial designs that were created, were followed and only 1 major change was made from the original designs. The majority of the time during this stage was spent implementing the data download services.

3.2 IDEs that were used

In this section I will discuss the two IDEs that were used whilst developing and the reasons why I switched from Eclipse to Android Studio.

3.2.1 Eclipse with Android developer tools

When following the setup tutorials on the Android developer guide [13], the tutorial taught me how to install the Android developer tools (ADT) plugin for Eclipse [18]. I had already used Eclipse for other academic assignments so were already familiar with the basic user interface.

The ADT plugin added the ability to create Android application's to Eclipse. The plugin also provided a GUI interface for downloading Android SDK's, including all documentation. The ADT plugin also allowed Eclipse to launch Emulators for a variety of devices.

As Eclipse is an IDE, when writing Android code with the ADT plugin installed code suggestions and syntax errors are shown in real time. Using an IDE greatly improved the speed at which I wrote code. As I was new to Android development the documentation suggestions provided whilst coding were extremely useful.

Eclipse is occasionally buggy when running on my computer, for example occasionally every line of code within the project will be highlighted as an error and I will therefore have to restart Eclipse for the error to disappear. Due to this error and a few other minor problems with Eclipse I searched the net for an alternative IDE half way through implementation.

3.2.2 Android Studio

Android Studio is an IDE created by the Android developers at Google [14]. Android Studio, which is currently still in beta is based off the IntelliJ IDEA IDE [14]. Although Android Studio is still in beta [14], the application works flawlessly, for everything that I needed it for.

As Android Studio was built solely for Android development it does not contain un-wanted bloat like Eclipse. This makes it easier to find what you want in the menus as they contain fewer options. Android Studio also has more intelligent code suggestions than Eclipses, making development easier. Android Studio build configurations are configured using Gradle, which meant that adding library's and custom build configurations was easy within Android Studio [14].

After running into issues with Eclipse I found the early access preview of Android Studio on Android developer website. Installing Android Studio was easy and the SDK's and emulators I had downloaded within ADT were automatically transferred. Android Studio also provides an automatic migration manager to convert projects from Eclipse into Android Studio. I migrated the main application into Android Studio, but never migrated the projects prototypes, as they were already complete at this stage in development.

3.3 Use of Gradle

Grade [26] is an automatic build configuration tool, similar to Apache's Maven. Gradle allows you to create custom build configurations [26] for a variety of setups. Gradle also has support for handling application dependencies, meaning importing and managing external libraries is easier and more efficient than manual methods.

I setup two build configurations using Gradle. The first configuration was used for building the application so that it was ready to be debugged. This configuration automatically enabled the debugable option within the Android manifest configuration file and also disabled Android's ProGaurd [37]. Enabling the debugable variable allows the application to be ran in the Android Studio debugger and disabling ProGaurd prevented the built APK from being obfuscated and shrunk [37], which improved the build time.

The second build configuration that was setup was the configuration for building the release version of the application. This build setup disabled the debuggable options and enabled Android's ProGaurd. Enabling ProGaurd prevents other developers from easily stealing your code as the generated code is obfuscated. Using ProGaurd also shrinks the file size and optimizes the final APK [37].

On both of the above-mentioned build configurations I used Gradle to automatically check for the latest version of Robospice and download it if needed, including all of the needed dependencies. I found this feature extremely useful as Robospice had a list needed dependencies, that'd have to have been managed manually.

3.4 Test driven-development

I had originally planned to follow the test-driven development pattern [28] taken from the eXtreme programming [2] methodology throughout the project. As this was my first ever large project that I had used test-development I found using that it slowed down my development. As this project had a limit time scale, I decided to change the original plan of using test-driven development for the entire project to only using test driven development for critical classes such as the calculator and view drug activity. I then followed the waterfall model's method of testing for the remaining classes, which is to test the application thoroughly once implementation was complete.

3.5 Implementing the models

The first package that I implemented for the project was the models package. The model's had been first implemented within the prototypes, but the models had been slightly updated during the design stage, so these changes were made.

Implementing the models first allowed me to have classes to contain the data used for the system, meaning that classes that use the models will be able to use them.

3.6 Database helper class

Once the user models had been created, a database to store the models was implemented. The database helper class manages the database structure (creating and upgrading). The class also provides the rest of the classes with the ability to insert and retrieve data from the database. The implementation of this class went as expected and the class works excellently. Static variables were used throughout the class to allow the database to be customised easily.

3.7 Authenticating the user

The next part of the system to be implemented was the login activity and the authentication class. Implementing the authentication would allow the user's username and password to be stored for the downloading of data, so this was logically the next step to implement.

When starting this project the only method of authenticating a user was to request a piece of data from the provided API and checking if an error was thrown, therefore I used the drug indexes API URL as this was the smallest XML file [49] to download. Although I used the smallest file possible whenever the login was successful the request would take several seconds as the drug index was downloaded. As the drug indexes were not used when logging in, this was a waste of data. To improve this I emailed the NHS representative and asked them to create a new API URL for authenticating.

The NHS [34] implemented the newly requested API URL. This URL takes two parameters (the username and password of the user) and returns true or false if the credentials are correct. This new API URL greatly sped up the login process and improved efficiency.

3.8 Downloading the data and populating the local database

The next logical step was to implement the classes for downloading the database. Implementing the database and populating it early on ensured that all classes that used the data could be implemented afterwards.

This is the section of the implementation stage that changed majorly from the initial design. In the original design I planned to carry out the task of downloading the data using AsyncTasks. As mentioned in the design chapter of this report, AsyncTasks were not appropriate for long running tasks [17] as they're attached to the activity. This meant that if the user minimised the application or changed the devices orientation the download would be cancelled. As the tasks of downloading all the data was not a short running task and that I wanted the user to be able to run the task in the background I could not use AsyncTasks.

As my original design would not work how I had expected I had to redesign the download classes. I then learnt about Robospice services [39], which would allow me to carry out the download in the background. Robospice services run in their own thread [39] and therefore the download's can be ran simultaneously through multi-threading.

Whilst using Robospice services I still encountered some problems. Robospice was not built for downloading data and storing it within the database, Robospice was built for long running HTTP requests [39] such as downloading large images from the web. Due to the intended nature of the Robospice services the caching abilities of Robospice did not suit my application. This is because Robospice only caches the return value of the service, as my application adds the information to the database within the service, there is no return value. After researching into the best practices I learnt that Robospice has a class made specifically for tasks that are un-cacheable services [39], thus this class was extended in all of the applications services.

Another issue I encountered when implementing the download service was determining when all the services had finished downloading. If the user had the application open whilst the download was in progress the download task worked as expected as the on success and on failure methods of the activity were called, but if the user minimised the application and a service completed the task, when the service attempted to call the on success or on failure method nothing would occur as the activity would not exist at that point.

To solve this issue, extra methods had to be added to the DataProgress singleton, these extra methods keep track of the amount of started services and the number of completed services. To keep a track of the number of completed service I had to read into the Robospice service source code [39] and plug into the method that notifies the activity when a service is complete, I then override this method and increased the finished count within the DataProgress singleton. When the number of started services is equal to the number of completed services then all the services have finished. The applications then checks that all the required API URLS have been downloaded, if they haven't the user is notified of the failure and given the option to retry to download the parts that failed.

The finished implementation of the download activity and service works excellently, both in the background and in the foreground. The user is notified of any errors, even if the errors occur in the background. If errors occur whilst in the background, when the user reopens the applications they are presented with the retry option. Whilst the download service is in progress a notification is placed into the notification centre, which allows the user to know the download is occurring.

3.9 Main, browse and view drug activities

Once the data download had been implemented it was possible to download the full set of data and store it within the applications local database. The next step was to display this data onto the screen. To achieve this the mock-up designs were implemented into the application for each of the views. Once user interface for each of the activities had been implemented the user interface was then populated using the data from the local database.

The browse drugs page displays a list of all the drugs, which is taken from the drug indexes table. The user can enter text into the search box to filter the results within the list. Android ListAdapter provided an easy to use method for filtering the results automatically by just passing the filter text as a parameter to the filter method.

Once the browsing of drugs had been implemented the next step was to implement the view drug page, a prototype of this page had already been made, so the prototype was copied into the project and edited to work with the new drug model. The implementation worked effortlessly and the drug data was displayed on the screen.

There was a slight aesthetic issue with the displaying of data. The data provided from the API URL contained HTML. The native Android TextView object only has basic HTML support. The issue arose due to the references within the provided HTML using SUP tags. SUP tags in HTML are the tag used to super-script a piece of text, Android TextView HTML does support super scripts but the super scripts cause the line spacing to increase for lines that use super scripts. This made the drug's information look misaligned which wasn't aesthetically pleasing.

To solve the aesthetic issues within the HTML I implemented a pre-parser to parse the HTML and replace all SUP tags with SMALL tags. Using SMALL tags made the references smaller, which made them standout, without affecting line height. I decided to execute the parsing during the displaying of the data rather than during the data download. Although this may not be the most efficient way of parsing the data, it ensures that the database contains an exact copy of original database.

As the prototype did not support the drug information header helpers this had to be implemented. An icon from the Glyphicon's open source library [25] was edited to match the colour scheme of the application. The icon was then added to the project and displaying of the icon where needed was implemented. To track which icon or header had been clicked Android's tag feature was used, this allowed each header and icon to contain a tag of its ID within the database. When a user clicks on the header the helper information corresponding to its tag is opened into an alert dialog.

3.10 Calculator implementation

The next classes to be implemented were the calculator classes. I asked the NHS representative for information on the calculations and their equations during the first email I sent, they replied that they would be able to provide the information within the following week. This information had still not been received when the project had reached the stage in which the calculator would be implemented. Therefore I sent a further email to the representative asking for this information, as the representative was not in work at the time, they provided an excel spread sheet containing the data needed for the calculations and the C# code that they used for the calculations.

The C# code was then read to derive the equations used for the calculations. Once the equations for calculation infusion rate from dosage and dosage from infusion rate were known the applications calculator and calculator activity class could be designed.

The data within the excel spread sheet was then converted into an XML file [49], this XML file was then temporarily added to the project and the data parser for it was added to the data downloader. The NHS representative was then emailed, requesting them to implement an API URL for retrieving the data in the desired format. This allowed the development of the project to continue whilst the data was not accessible. The NHS representative later implemented the API URL and the calculator XML file was removed from the project.

As this class outputs information that could be lethal if incorrect a test-driven development [28] approach was used whilst implementing it. As test data was needed, the NHS's Medusa website [20] was opened and a variety of calculations for multiple drugs were performed. The test data was then written into the unit tests, which would later be used to test the calculations.

The first method of the calculator to be implemented was the validation method. The purpose of this method is to return an integer value that represents the result of the validation. The possible integer values are stored as public static variables, meaning other classes can access their values, which is used when checking the result. The implementation began by creating test data that would return each of the available return types, such as success, invalid values and warnings. Once the test data had been written, the code to allow all the test cases to succeed was written.

Once the validation method had been successfully implemented, the calculate method had to be implemented. As the test data for the calculator class had already been gathered the implementation to make the unit tests successful was the next step.

After every unit tests was successful the implementation of the calculator class was complete. The next class to be implemented was the calculator activity. The calculator activity class is responsible for providing the view to the user so that they can enter the information for the calculation. The design of the calculator activity allowed the user to select the type of calculation and the needed and un-needed fields were showing or hidden from the view. Hiding the un-need input fields helps improve the usability of the application. Once the user has entered the required information and clicked the calculate button, the information entered is validated. If the information passes validation the calculation is shown, otherwise the error is displayed using an Android Toast or a warning dialog is displayed if a warning is thrown [8]. The design of the calculator activity was implemented successfully and activity works as expected.

3.11 Showing the calculation performed

It had been decided that the result of the calculation would be displayed within a dialog [8]. The dialog had been designed so that the result of the calculation and the equations used to perform the calculation would be shown. The original plan was to display the equations within a TextView and use HTML to format the equations correctly.

It was decided that the HTML horizontal row (HR) tag would be used to display a horizontal line separating the numerator and denominator of the calculator equations. As Android TextView's HTML support is limited, the HR tag is not available within TextView's and therefore could not be used to display the calculation equations and results.

A WebView was used within then dialog to neatly present the user with the equation used and the results of the calculation. WebViews have larger HTML support than TextViews, they support both HTML and CSS. The calculation was displayed using HTML and then formatted using CSS. Relative font sizes were used within the WebView to ensure the result would be the same across devices. Once the WebView is displayed onto the screen the WebView is zoomed so that the HTML fills the dialog.

An extra benefit of using a WebView is that WebView's allow the user to zoom in and out using a pinching motion with the fingers. This allows the user to make the equations and result smaller or larger should they want to.

The finished implementation successfully presents equations used and the result of the equations to the user. This allows the user to verify the correct calculation has been performed.

3.12 Adding access to the calculator

Once the calculator classes had been created, a method of allowing the user to access the calculator was needed. To achieve this, the view drug activity was edited so that when a calculator is available for a drug, a button is displayed to open the calculator.

As only a limited number of drugs contain calculators I wanted to create a simple method for finding drugs that contain them. An activity similar to the browse drugs activity was created. This activity allows the user to search through a list of drugs that contains calculators. This activity speeds up the process of opening the calculator for the user.

3.13 Extracting strings from the Java code

Within Android you can define string variables within an XML file called strings.xml [47]. By placing all strings used throughout the application within this file improves the maintainability and robustness of this system, as should future developers ever need to change the text contained within a string, they will only need to change the string within one file. Also using the strings.xml file allows future developers to easily provide extra language support [47].

Once the applications functionality was complete, code refactoring was executed to ensure that the code was efficient and easy to maintain. Whilst refactoring, all strings within the application's code were extracted and placed into the strings.xml file [47]. This will allow the NHS to quickly modify the text throughout the application, without needing to learn Android development. If the NHS would like to support the Welsh language within the application, which they may want to do, as the application was produced for NHS Wales, they only need to create a new directory for Welsh language support and then translate the strings.xml file into Welsh.

3.14 Implementing XML customisability

Early on in the projects lifecycle the NHS mentioned that they have multiple sets of data in a similar format to the data used for this application. They also mentioned that they plan to create multiple applications for the various data sets. The NHS asked for a simple method of modifying this application to allow them to create multiple applications from the other datasets. Throughout the design and implementation stages of the project this request was considered but not initially implemented, as it was an additional extra, providing there was sufficient time.

As the implementation stage of the project ran as planned, there was enough time to implement the NHS's request. To implement this functionality an addition XML file similar to the strings.xml [47] file was created, called data_download.xml. The purpose of the data_download.xml file was to provide the API URLs at which the XML files containing the data could be requested. The data_download.xml file also contains the XML tags that relate to database tables.

The API URLs provided within the data_download.xml may contain two parameters, %USERNAME% and %PASSWORD%. The application will automatically replace these parameters with the users saved username and password. This allows the URLs to be changed by someone with very little programming experience.

To download the drug and drug information the application currently uses 26 URLs, one for each letter of the alphabet. Hard coding the URLS of the 26 sets of data would be bad software engineering. To accommodate for the multiple URLs, a list of the URLs and tags for them were added to the data_download.xml file. The tags are used as a description of the URL that can be displayed to the used to relay feedback to the user, for example if an error occurred whilst downloading letters beginning with A, the application will alert the user by outputting Failed to download letters beginning with A, the tag in this case would be letters beginning with A.

In order for the application to parse the data correctly, the XML tags [49] used within the data are described inside data_download.xml. These descriptors include the tag name of the repeating element, for example the XML signifying the start of a new drug. The XML descriptors also include the tag names that contain the pieces of information that will be mapped to the database tables. By knowing the name of the repeating elements and the name of the tags containing the information within the repeating elements, the applications can populate the local database,

With both the customisability of the strings.xml file and the data_download.xml file, a developer with no experience of Android development will be able to create multiple applications from varying API URL's effortlessly.

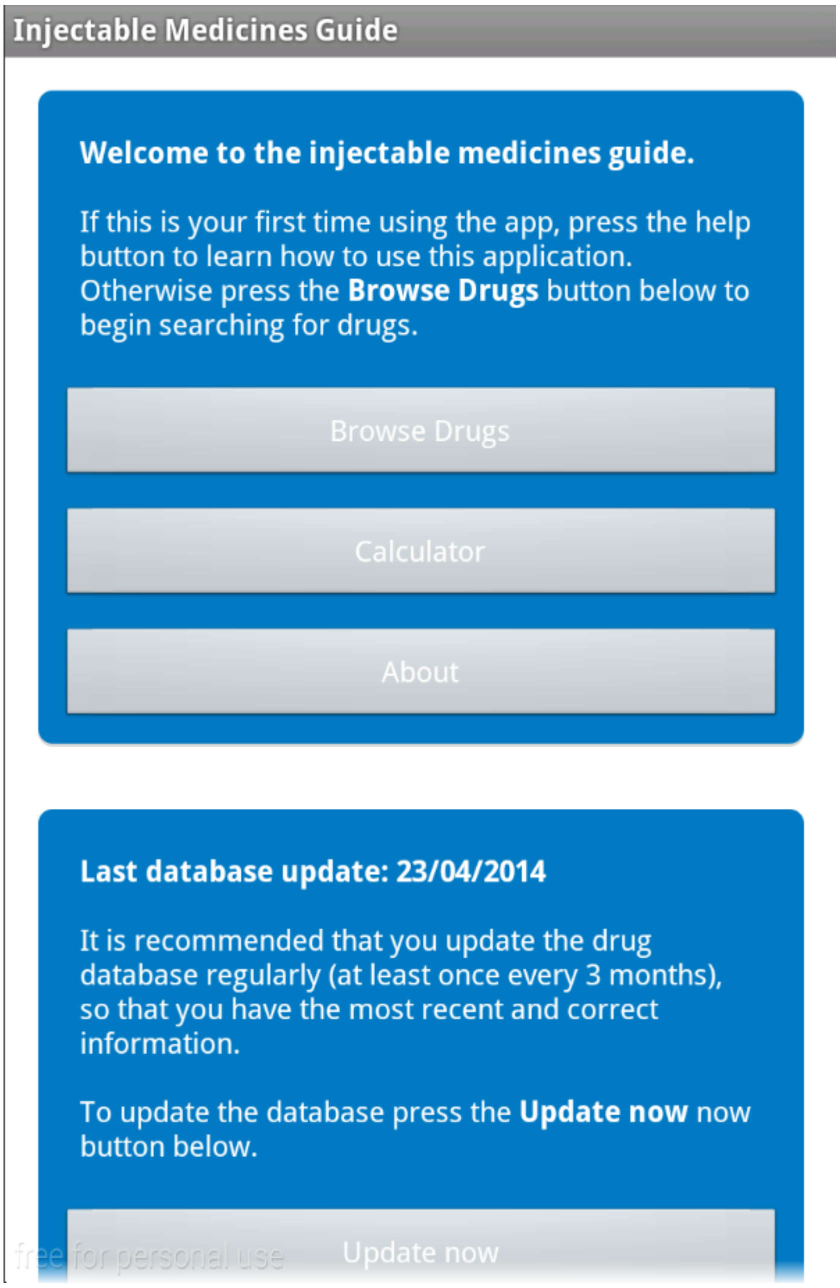
3.15 Supporting older devices

To improve the available reach of the application, it must support the earliest version of the Android SDK [13] as possible. The minimum SDK required for the libraries and dependencies that were used was API version 8 [39]. API version 8 (Froyo) was released on 20th May 2010 [21]. The majority of Android users are currently on API version 8 or greater, it was therefore decided that the application must support all versions above API level 8 [10].

During the implementation of the application a device running API version 19 was used. Once the implementation was complete, the minimum SDK version of the application was changed to API level 8 and ran on a device running Froyo [21]. The application ran as it would on a newer device, only issues with the user interface were found.

When the application ran on the older device the custom colour scheme was not seen, this is because the customisation of the ActionBar was added at a later API version. Although the colour scheme was not seen, the application was still aesthetically pleasing.

On the older device the transparency of the buttons within the MainActivity was not applied, because of this the button was displayed, as a white button with white text thus the buttons text was not visible on the device. A new layout for the MainActivitiy, specifically for device earlier than API version 10 was created to fix the styling issue on the device.


Injectable Medicines Guide

Welcome to the injectable medicines guide.

If this is your first time using the app, press the help button to learn how to use this application. Otherwise press the **Browse Drugs** button below to begin searching for drugs.

Last database update: 23/04/2014

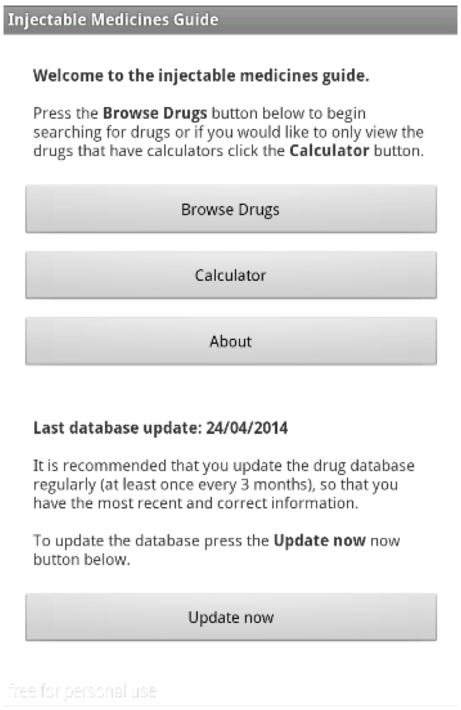
It is recommended that you update the drug database regularly (at least once every 3 months), so that you have the most recent and correct information.

To update the database press the **Update now** now button below.

Figure 3.1: Main activity on API 10 before improvement.

3.16 Completed application screenshots

This section includes screenshots of the final application, running on two devices, one running API version 10 and another running API version 18. Android's style changed greatly between these two versions, the screenshots prove that the application is aesthetically pleasing on both devices.


Injectable Medicines Guide

Welcome to the injectable medicines guide.

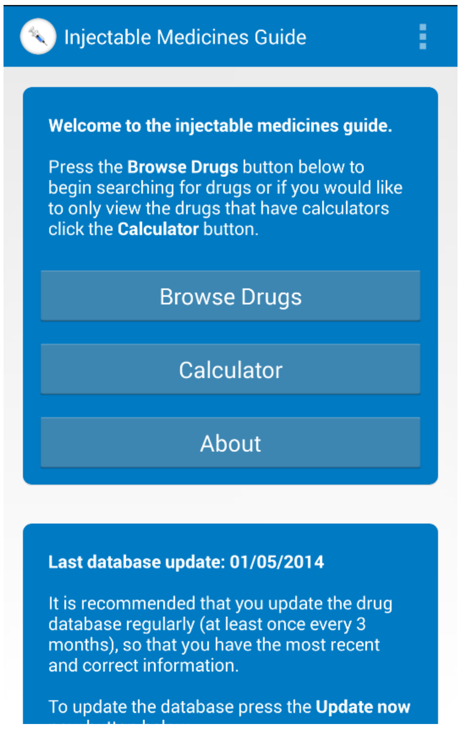
Press the **Browse Drugs** button below to begin searching for drugs or if you would like to only view the drugs that have calculators click the **Calculator** button.

Last database update: 24/04/2014

It is recommended that you update the drug database regularly (at least once every 3 months), so that you have the most recent and correct information.

To update the database press the **Update now** button below.

Figure 3.2: Main activity on API 10


Injectable Medicines Guide

Welcome to the injectable medicines guide.

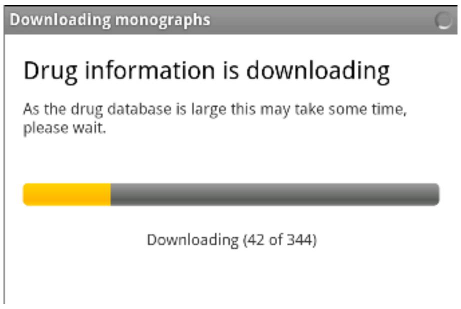
Press the **Browse Drugs** button below to begin searching for drugs or if you would like to only view the drugs that have calculators click the **Calculator** button.

Last database update: 01/05/2014

It is recommended that you update the drug database regularly (at least once every 3 months), so that you have the most recent and correct information.

To update the database press the **Update now**

Figure 3.3: Main activity on API 18

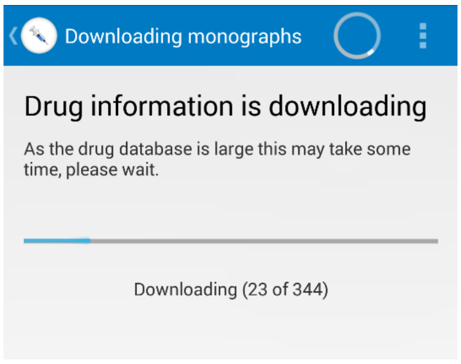

Downloading monographs

Drug information is downloading

As the drug database is large this may take some time, please wait.

Downloading (42 of 344)

Figure 3.4: Download activity on API 10

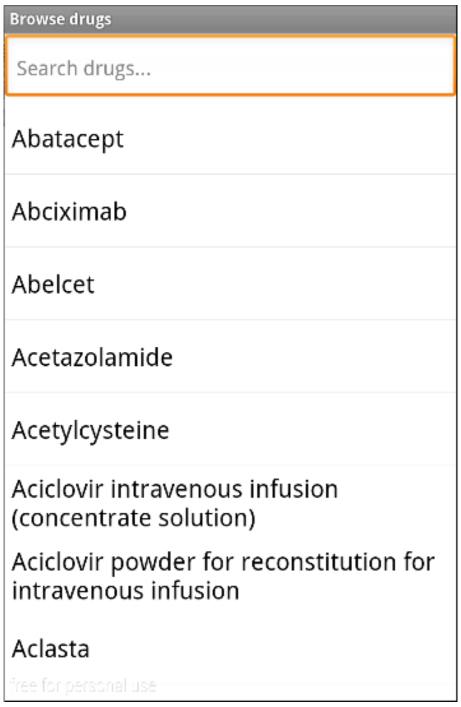

Downloading monographs

Drug information is downloading

As the drug database is large this may take some time, please wait.

Downloading (23 of 344)

Figure 3.5: Download activity on API 18


Browse drugs

Search drugs...

Abatacept

Abciximab

Abelcet

Acetazolamide

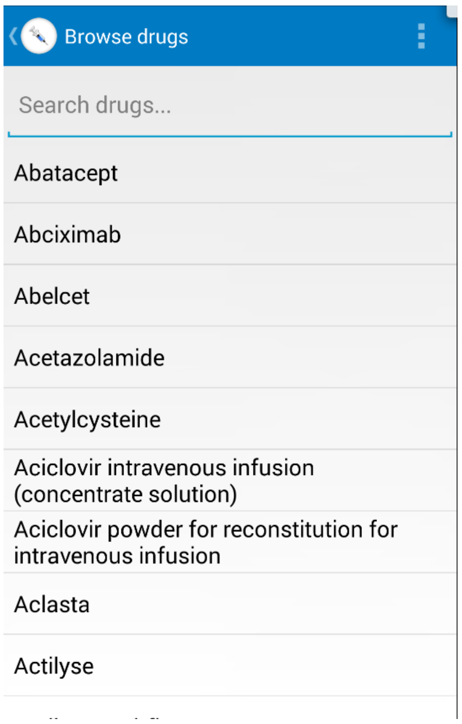
Acetylcysteine

Aciclovir intravenous infusion (concentrate solution)

Aciclovir powder for reconstitution for intravenous infusion

Aclasta

Figure 3.6: Browse activity on API 10


Browse drugs

Search drugs...

Abatacept

Abciximab

Abelcet

Acetazolamide

Acetylcysteine

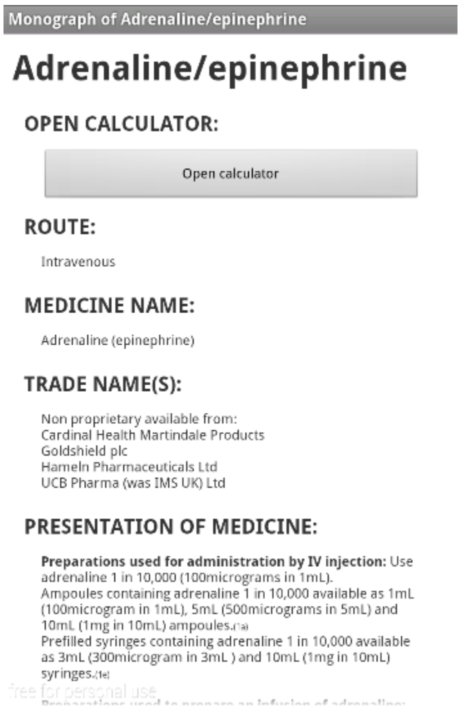
Aciclovir intravenous infusion (concentrate solution)

Aciclovir powder for reconstitution for intravenous infusion

Aclasta

Actilyse

Figure 3.7: Browse activity on API 18


Monograph of Adrenaline/epinephrine

Adrenaline/epinephrine

OPEN CALCULATOR:

Open calculator

ROUTE:

Intravenous

MEDICINE NAME:

Adrenaline (epinephrine)

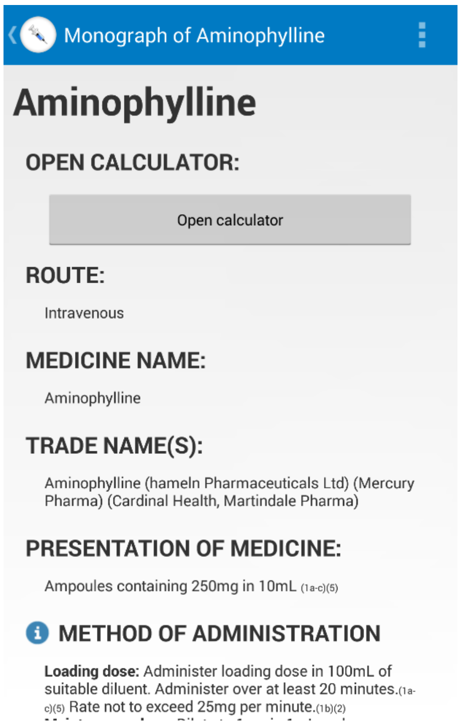
TRADE NAME(S):

Non proprietary available from:
Cardinal Health Martindale Products
Goldshield plc
Hameln Pharmaceuticals Ltd
UCB Pharma (was IMS UK) Ltd

PRESENTATION OF MEDICINE:

Preparations used for administration by IV injection: Use adrenaline 1 in 10,000 (100micrograms in 1mL).
Ampoules containing adrenaline 1 in 10,000 available as 1mL (100microgram in 1mL), 5mL (500micrograms in 5mL) and 10mL (1mg in 10mL) ampoules.(1a)
Prefilled syringes containing adrenaline 1 in 10,000 available as 3mL (300microgram in 3mL) and 10mL (1mg in 10mL) syringes.(1e)

Figure 3.8: View activity on API 10


Monograph of Aminophylline

Aminophylline

OPEN CALCULATOR:

Open calculator

ROUTE:

Intravenous

MEDICINE NAME:

Aminophylline

TRADE NAME(S):

Aminophylline (hameln Pharmaceuticals Ltd) (Mercury Pharma) (Cardinal Health, Martindale Pharma)

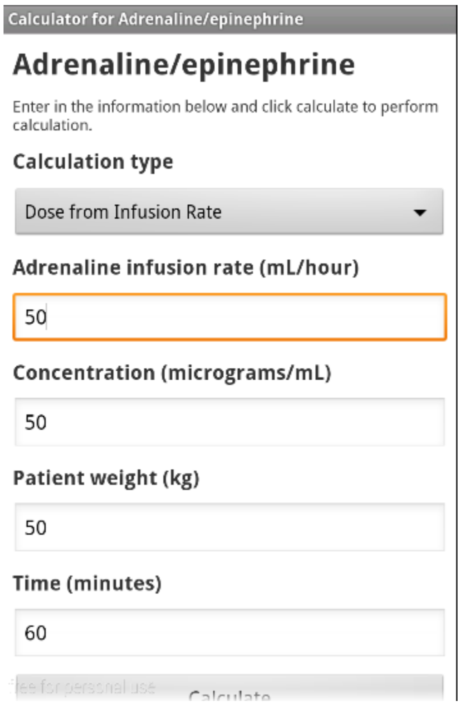
PRESENTATION OF MEDICINE:

Ampoules containing 250mg in 10mL (1a-c)(5)

METHOD OF ADMINISTRATION

Loading dose: Administer loading dose in 100mL of suitable diluent. Administer over at least 20 minutes.(1a-c)(5) Rate not to exceed 25mg per minute.(1b)(2)

Figure 3.9: View drug activity on API 18



Calculator for Adrenaline/epinephrine

Adrenaline/epinephrine

Enter in the information below and click calculate to perform calculation.

Calculation type

Dose from Infusion Rate

Adrenaline infusion rate (mL/hour)

50

Concentration (micrograms/mL)

50

Patient weight (kg)

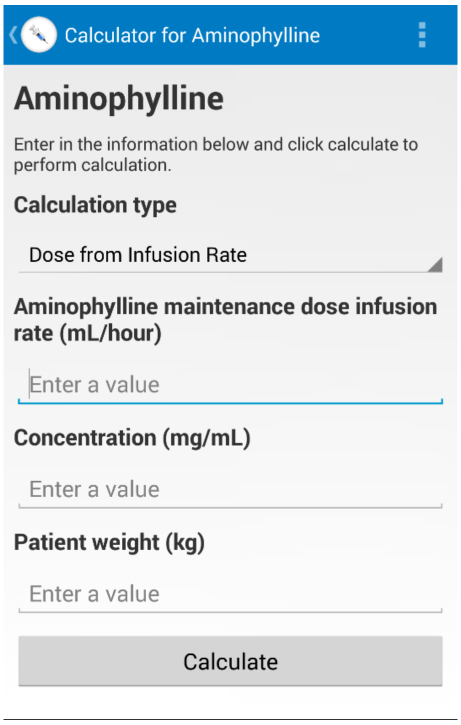
50

Time (minutes)

60

Calculate

Figure 3.10: Calculator activity on API 10



Calculator for Aminophylline

Aminophylline

Enter in the information below and click calculate to perform calculation.

Calculation type

Dose from Infusion Rate

Aminophylline maintenance dose infusion rate (mL/hour)

Enter a value

Concentration (mg/mL)

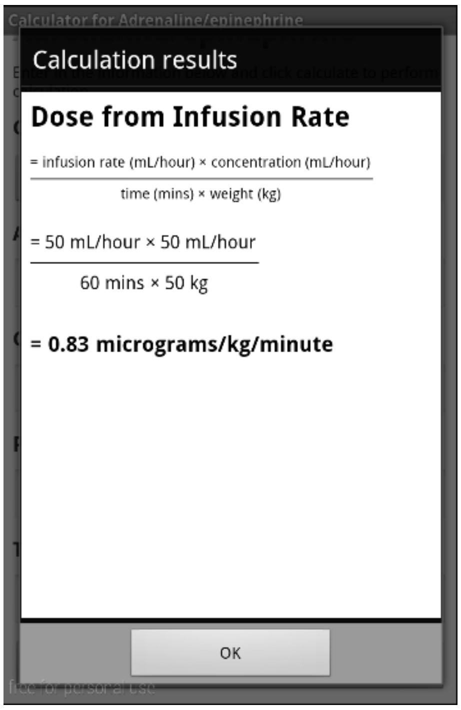
Enter a value

Patient weight (kg)

Enter a value

Calculate

Figure 3.11: Calculator activity on API 18



Calculator for Adrenaline/epinephrine

Calculation results

Dose from Infusion Rate

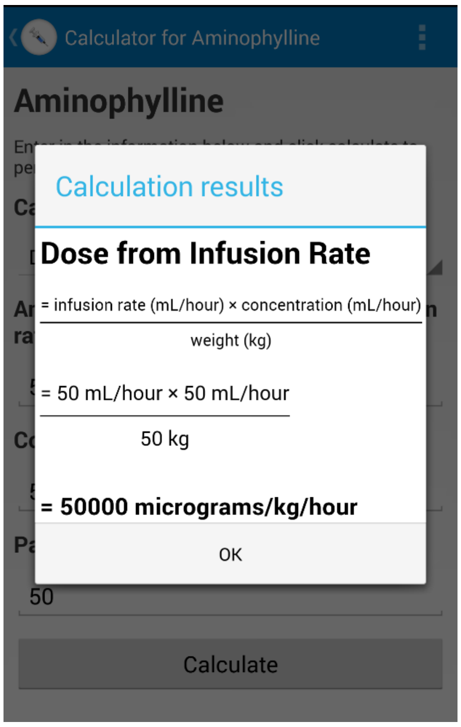
$$= \frac{\text{infusion rate (mL/hour)} \times \text{concentration (mL/hour)}}{\text{time (mins)} \times \text{weight (kg)}}$$

$$= \frac{50 \text{ mL/hour} \times 50 \text{ mL/hour}}{60 \text{ mins} \times 50 \text{ kg}}$$

= 0.83 micrograms/kg/minute

OK

Figure 3.12: Calculator result on API 10



Calculator for Aminophylline

Aminophylline

Calculation results

Dose from Infusion Rate

$$= \frac{\text{infusion rate (mL/hour)} \times \text{concentration (mL/hour)}}{\text{weight (kg)}}$$

$$= \frac{50 \text{ mL/hour} \times 50 \text{ mL/hour}}{50 \text{ kg}}$$

= 50000 micrograms/kg/hour

OK

50

Calculate

Figure 3.13: Calculator result on API 18

Chapter 4

Testing

Testing is extremely important to the success of any software development project [28]. A thorough testing strategy increases robustness and customer confidence within an application. As the application is used to deliver information that will be used by medical staff, it is vital that the application is well tested, as the consequences of poor testing could be fatal. Testing is one of the fundamental steps within the water model [3] and therefore by choosing this methodology testing was enforced. To provide details of the testing carried out during this project a test specification document was produced. This chapter will discus the approach to testing used throughout the project and also describe test strategies used in detail.

4.1 Approach to testing

Testing was used throughout the implementation process to help create a robust and effective application. Originally it was planned to use a test-driven development [28] throughout the entire implementation process, but this was later found to be detrimental to the progress of the application. Although test-driven development was not used throughout the implementation process it was used when developing classes that could potentially be fatal, such as the calculation classes.

Throughout development the application was ran on a real device and any new features were thoroughly tested, using a variety of behaviours and carefully monitoring the applications state for any anomalies. This approach to implementation allowed for a well-tested application, before any test strategy had been complete.

After the implementation phase of the project had been complete, thorough testing of the application was executed. Unit tests [31] were written for all classes that had not already had unit tests created. User interface tests were created and executed using android activity unit tests [6]. The application was also stress tested by using Android's exerciser's monkey application [7].

Once the tests had been written, several devices of varying setups were created using the Genymotion emulator [22]. The full list of tests were then executed on each individual device. This ensured that the application runs well on an array of devices.

4.2 Test database

When the testing process began an extra parameter was added to the database constructor, this parameter would allow the application to open a separate database that was identical to the actual database, this separate database could then be used during tests. Having a separate database whilst testing meant that the data could be manipulated without effecting the main application. This was needed, as downloading a new set of data after each test would take a long amount of time.

4.3 Unit testing

The Android SDK [13] provides classes for unit testing your application. These classes are based off the JUnit framework [31], they add extra features such as the ability to access the applications context allowing you to test the database of the application.

Unit tests [31] were written for every class of the application, testing each public and protected method. Most tests contained multiple assertions, testing that the expected output was returned when correct information is entered and that an error is raised when the incorrect data is entered.

As the unit tests have now been written for all classes, should a developer in the future make any changes to the application, they can execute the unit tests to ensure they have not broken anything. As the unit tests will be provided to the NHS with the source code, they will be able to execute the unit tests for themselves, allowing theming to verify thorough testing was carried out, thus increasing their confidence within the application.

As Android runs on a large amount of devices [10], unit tests are useful for quickly testing that the application runs as expect across the Android platform.

4.4 Bug that was found due to unit testing

Test data had been gathered for testing the calculator class. This data was then implemented into unit tests. To ensure the class was well tested a wide spectrum of values were used, including both small and large values.

Within the original implementation of the calculator class I had used floats to store all values and results of the calculation. After executing the unit tests, most tests were succeeding, but few were returning values slightly off the expected value. As the dosage given to patients must be correct to good level accuracy I began investigating the cause of the problem.

It was found that the issue lay with the data types used, the NHS's website [20] used doubles whereas I had used floats. Due to doubles being more accurate than floats, the data types were changed to doubles. The unit test were then executed again and all tests succeeded.

4.5 Automated user interface testing

Android's activity tests allow you to comprehensively test the user interface of the application [6]. Activity tests simulate the launching of the application's activity [6] and perform a set of unit tests on that activity. These activity unit tests can test a variety of user interface values.

Activity tests were created for every activity. Within the tests each element is checked to ensure that the element is displayed on the screen, the width and height of the element as set correctly, and if the element contains text that the text is set correctly.

Activity tests can test whether an element is visible to the user at that current moment, for example an element may not be visible when the element is displayed at the bottom of a scrollable view. On both the browse drugs and browse calculator activities the search box must be visible to the user at all times. Activity tests were created to ensure the search box is visible to the user.

When the user changes the calculation type on the calculator activity, the dosage and infusion rate fields are toggled, depending on the selected option. As well as this, when a drug calculator is opened for a drug that does not require time or patient weight, the appropriate fields must also be hidden. Activity tests [6] were created to test the calculator activity under a variety of situations to ensure that only the correct fields were displayed.

It was very important to run the activity tests on a variety of devices, this ensured that the user interface functions the same on all devices, without having to manual test each device's user interface.

4.6 Automated integration testing

Due to Android security policies you cannot test the functionality of a user interface element by using activity tests [6], for example when running an activity test you cannot simulate the user entering text into a text field. Robotium is an automated black-box testing framework [40] for Android licenced under the Apache 2 licence [15]. Robotium allows a developer to implemented black-box integration tests using code [40]. There is also a commercially available version of Robotium, which will record user interface actions and monitor the results [40], then automatically generate the Robotium code.

Although automated black-box testing using Robotium would be useful for the project, it was decided that automated tests would not be needed for a project this size. The amount of time spent learning Robotinium and then implementing it, would've taken longer than manual executing the integration tests. If the amount of activities and the functionality within them increase, Robotinium tests should be implemented.

4.7 Integration testing

As Robotinium was not used, a method of testing that the combined units of the application work as expected was needed. To test that the application works as expected a list of integration tests were made and then ran on two devices, one device running API version 19 and the other running API version 8 [21]. Using two devices, with varying ages increases the accuracy of the results. To further increase the accuracy of the results the tests would be executed on more devices.

A full list of the integration tests made can be found in the integration testing section of the test results appendix.

4.8 Automated stress testing

Exerciser Monkey [7] is a tool provided with the Android SDK [13], which is used for stress testing Android applications. The tool simulates a set amount of random events on the device, such as button presses, screen presses, volume changes and screen rotations. The tests are used to ensure that applications run well under stressful tasks and that parts of the application do not throw errors.

It was planned that the application would be installed on a device, then using Exerciser Monkey, execute 5000 random events to the device. The events were executed and the application never crashed, but whilst watching the device I noticed that the exerciser monkey randomly clicked the logout button, from then on the only activity that was tested was the login activity, as it was impossible for the random events to enter a correct username and password.

It was important for all areas of the application to be tested using this stress tester. Therefore the logout functionality of the application was disabled, which was easy due to modular structure of the application. The Exerciser Monkey was then executed again, simulating 5000s events. The application handled the stress tests well and no errors or timeout warnings were displayed within the console.

4.9 Acceptance testing

To test that the application was at an acceptable level, acceptance testing was carried out. Before sending the application to the NHS for their approval it was vital to test that the application met all of the original functional requirements. This ensured that the application created achieves all of the original goals.

Table 4.1: Table of acceptance testing the functional requirements

	Functional requirement	**PASS/FAIL**
FR 1	**Authentication**	
FR 1.1	User must be able to authenticate themselves using their credentials	PASS
FR 1.2	User must be notified if the password they enter is incorrect	PASS
FR 1.3	User will be notified if the authentication failed due to connection issues	PASS
FR 1.4	User must be able to logout of the system, removing all data	PASS
FR 2	**Database synchronisation**	
FR 2.1	After login or when the user presses update the application must truncate all database tables and begin downloading new data	PASS
FR 2.2	Download complete list of drug indexes from database	PASS
FR 2.3	Download complete list of drugs and drug information's	PASS
FR 2.3	Download all information needed for calculating doses and infusion rates.	PASS
FR 2.4	Download must still run when the application is in the background	PASS
FR 3	**Menu options**	

FR 3.1	Upon pressing the Menu button on the device the user will be presented with a list of available options, which execute tasks (Logout, exit, search)	PASS
FR 4	**Main screen**	
FR 4.1	Upon successful data download the user will be displayed with a screen where they can navigate to other parts of the application	PASS
FR 4.2	User will be see when an update was last performed, and perform an update from this screen.	PASS
FR 5	**Browse drugs**	
FR 5.1	This screen will allow the user to view a list of all drugs	PASS
FR 5.2	There will be an input box on this screen, when the user enters text into the input box the results in the list will be filtered to only show results related to the input	PASS
FR 5.3	The user will be able to click a drug in the list to open a new screen displaying the needed information	PASS
FR 6	**View drug**	
FR 6.1	When a drug has been selected the drug and all it's information will be displayed in an easy to read format	PASS
FR 6.2	Where drug information headers contain help information, a help icon will be displayed next to the header.	PASS
FR 6.3	When heading help icon is clicked the helping information will be displayed	PASS
FR 6.4	If the drug had calculator information, then a button to open the calculator should be shown	PASS
FR 7	**Browse drugs with calculators**	
FR 7.1	A view similar to the browse drugs view will allow the browsing of drugs that contain calculator information.	PASS
FR 8	**Calculate dose and infusion rate**	
FR 8.1	The user will be able to select calculation type	PASS
FR 8.2	User will be able to enter information required for the calculation	PASS
FR 8.3	When the calculate button has been clicked the input will be thoroughly validated	PASS
FR 8.4	After validation the result of the calculation will be displayed to the user	PASS
FR 8.5	The equation and values used to calculate the answer will be neatly displayed to the user	PASS
FR 9	**XML customisability for developers**	
FR 9.1	All text within the application must be changeable through XML files.	PASS
FR 9.2	The structure of the XML API's provided must be outline within XML files, allowing easy customisation for different API's	PASS

As shown in the table above the application fulfilled the complete list of functional requirements. The next logical step was to send the compiled APK to the NHS representative for approval.

The representative was very pleased with the final application and only suggested one change, which was to order the calculator's in alphabetical order. This change was implemented and an updated sent to the NHS. Once the changes had been made, the NHS representative demonstrated the application to the IMG Executives Group that consisted of pharmacists and nurse experts, whom stated that the application was intuitive to use and easy to navigate.

4.10 Acceptance testing feedback

This section contains the feedback provided by the NHS representative.

"For the first time this year, NHS Wales Informatics Service became involved with Aberystwyth University when we proposed the development of a mobile phone app for the NHS Injectable Medicines Guide (IMG) as a possible dissertation project. The IMG is a website that that provides information on how to prepare and administer injectable drugs and is used by over 90% of NHS Trusts across the UK by nurses, doctors and pharmacists. It is a major contribution to patient safety and is recommended by the National Patient Safety Agency. We have been under pressure to develop a mobile phone app version of the website.

We supplied an outline design of the app. Aidan was one of two students to select this proposal to take forward as his dissertation. I am providing this feedback as a result of working with Aidan while he has written the software.

My contact with Aidan has been via email and he has always responded promptly to my questions and provided full and accurate answers. Aidan has had to ask us to provide access to the various data sources and I suspect the access that we have provided has not been what he would have chosen but he has always been willing to adapt to what we have provided and work with the our timescales. He has only requested minimal information so he must have worked within his own initiative to complete his work.

And then he delivered the app for us to look at. I was impressed that he had considered how we would implement the software and provided a simple means for us to load the app. Initially we had problems loading the software but Aidan researched the problem and worked with us to resolve the issue which was largely due to the age of our equipment. Implementation and support is a different skill set to development and I was impressed by his problem-solving skills and his persistence in resolving the issue.

I was bowled over by the app that he has produced; it is well beyond our outline design. He has clearly had to adapt what was essentially an iPhone design to the Android platform and he has produced a design that is very clean and easy to use. He has clearly understood the user interface design principles and I much prefer his design of the pages and the navigation from one function to another over our design.

One aspect of the app is the ability to download and update the data from the central website and to manage the security of this function. Aidan has demonstrated that he understands the security issue and has devised an update that is simple to use but downloads the complex data sets *that are involved and has been able to transform the datasets into the information as it appears on the screen, matching across the keys to the each file.*

I have demonstrated the app to the IMG Executive Group which consists of Pharmacists from Imperial NHS Trust and an RCN nurse expert on injectable drugs and they were very impressed and found the app intuitive and easy to use. The view was that nurses, for whom very few apps exist and so are not expert users, would find welcome this app and find it easy to navigate.

Overall we are very impressed with the way that Aidan has interacted with us and with the product that he has produced. I am sure he will have an excellent career in software if this is the path that he chooses."

Robin Burfield, Development Manager, NHS Wales Informatics Service.

This feedback proves that the application completes its original purpose and that the user interface is clean and easy to use.

Chapter 5

Evaluation

This chapter will evaluate the completed project, discuss anything that would have been done differently and outline any future improvements that could be made.

5.1 Evaluation against original goals

The original goals of the application stated that a native Android application would be created to aid NHS staff in administering injectable medicines to patients. This goal has been achieved as a excellently-working native Android application has been created that, that has been approved by a representative of the NHS to be useful.

The next goal stated was that the application would be created using the standard user interface library, which is provided with the Android SDK [13]. Whilst implementing the user interface, no external libraries were used and only standard Android elements were used. This has enabled the final application to work on a large array of Android devices and to be instinctive for a user familiar with Android to use.

As users that may not be technically minded will use this application, the application therefore had to be easy to use. To achieve this the user interface was designed to be clean, simple and self-explanatory for basic users. Using the standard Android user interface library reinforced this goal [12]. The feedback left by the NHS confirmed that the final application is both intuitive and easy to use.

As the devices running the applications will not have a constant Internet connection, the application had to download a complete set of data to be used for offline use. The final application does complete this goal and the application can be used within airplane mode, without any degradation of functionality. At first I was worried about the size of the final application with the complete database, but this only amounts to 5 megabytes of disc space, which is very small in comparison to the space available on modern devices.

The downloading of data and storing that data to the local database had to be possible whilst in the application is in the background, as the download can take several minutes, especially on a mobile data network, the final application achieves this flawlessly. The user can navigate from the application and a notification is placed inside the notification centre alerting the user that the download is still in progress. If an error occurs whilst the application is in the background, when the user re-enters the application they will be asked what action they would like to perform next.

As NHS staff will be using the application to administer potentially lethal drugs, I wanted to ensure that the application worked as expected. Therefore I planned to thoroughly and vigorously test the application. Throughout the lifecycle of this project testing has been important. I believe the final application has been well tested and should work excellently on a multitude of devices.

The final original goal for the application was to only use publicly accessible libraries, this has been achieved and only two external resources have been used, both of which have been licences under the Apache software licence [15].

Overall I believe the application has completed all of the original goals to an excellent standard. I am very pleased with final application and believe I have created a robust, efficient and well-designed piece of software.

5.2 Evaluation against functional requirements

The requirements set out within the requirements specification were compiled with input from the NHS. Therefore for the application to be deemed successful it was vital that final application implemented the majority of the functional requirements. As shown in the acceptance testing section of the testing chapter, the final application satisfies all of the functional requirements set out in the requirements specification. I believe the finally implemented application has gone above the expectations set out by the original functional requirements.

I am pleased with the list of requirements first set out within the background and analysis stage of the project, they have not had to be modified at any stage in the project. I believe this to be due to thorough project analysis before the project begun.

5.3 Time management

As mentioned in the background chapter, a Gantt chart was created at the start of the project and followed throughout. I did take some time to begin the project due to other freelance work, but by the mid-project demonstration I had caught up with what had been planned. Afterwards the project continued at a steady pace and the project was finished in good time.

The Gantt chart helped motivate me to complete the application in good time, as I could see the tasks that needed to be complete, and how long I had left for each.

5.4 Design decisions

The implemented application follows the design created within the design specification very well. The design utilises the principles brought to the Java language [29] through object orientation and follows many of the software engineering best practices, such as using getters and setters instead of global variables. By using a modulated structure, modules created within this project can be extracted and used within another project with minimal modification. This will be useful if the NHS plan to create multiple applications.

The final implementation only varies slightly from the original design. This was due to deciding to use Robospice services [39] over AsyncTasks [17]. Due to this, instead of using one class for downloading the complete set of data, multiple classes were built to download and parse individual parts of the dataset. Using Robospice services added many benefits to the final application, such as allowing tasks to be executed in the background and to allow simultaneous downloads through multi-threading. They also improved the design of the application, as separating the downloading and parsing class into separate classes helped to separate classes, which avoided the implementation having a god object anti-pattern.

5.5 Methodology

The methodology used was adapted twice throughout the project. Originally a pure eXtreme programming [2] was planned, and then the methodology was changed to the waterfall model [3] with test-driven development [28] throughout and then later changed to be the waterfall model with test-driven development for critical parts of the system.

Reflecting on the project using the standard waterfall [3] method throughout would've been more effective for this project. I wanted to gain experience in using agile methodologies, but using a methodology that I had little experience in, was a hindrance to the project, as extra time was added.

The test driven development [28] that I executed did help in finding problems with the calculator class early on, but the problems would've still been found during the testing phase, at the end of the waterfall model.

5.6 Implementation

The implemented system is a well-engineered piece of software, which follows software engineering best practices well. The code is split into modules of similar classes and classes have been designed to be as simple as possible, meaning the code is well structured.

The stress tests shown within the testing chapter show that the application is robust and efficient, as no errors including time out errors were thrown whilst testing under a very large load.

The code style set out within the Android code style guidelines [11] has been used throughout the implementation. I have made one modification to this code style; I decided to prefix all private class variables with an underscore. This helped signify whether a variable is local to the method or local to the class.

The code is documented using JavaDoc [30] style comments, which has been compiled into a HTML document, allowing future developers to gain knowledge of the system easily [30].

The implementation allows a large amount of the applications to be modified by editing only XML files [49]. This means the NHS can easily customise the application, even if they do not have any native Android developers.

From the onset of the project I knew that I would not be the sole person to maintain this project and therefore planned to make the project easily maintainable. By using common software engineering principles, using a standard code style and providing extensive comments I believe I this has been achieved.

5.7 Changes to the implementation that could be made

Although the implemented application works as expected there is ways the implementation could be improved. These improvements mainly lie within the classes responsible for downloading the database.

When the application begins downloading the application first downloads the drug indexes, then the drug calculator information and then finally the drug informations are downloaded using multiple threads. When implementing the system, I believed this to be the best method of organising the download. If the application fails to download the drug indexes or the calculator information the applications has to wait for the user to decided what action they would like to perform. Another issue with this approach is that the indexes and calculation are downloaded sequentially and therefore Android's multithreading capabilities are not used, meaning the process may take longer.

To improve the download task, all services should be pushed to the Robospice service manager when the download activity starts. This will allow all the downloads to start simultaneously, using multiple threads. As the download progress (Download x drugs of y) requires the drug index to be downloaded first, a higher priority should be set for the index service and before the download has been finished a standardised message could be used, such as Download X drugs.

Another improvement would be to execute the drug letter SQL queries using transactions [44]. Currently drugs are added to the database as soon as they are downloaded. If the download fails whilst mid-way through a letter and the user retries the download, for the drugs that have already been downloaded, the application will attempt to add them again. The drugs are not added to the database, due to primary key constraints on the database, but I believe it is better practice to not have to rely on these constraints. Using a transaction to add drugs starting with each letter will prevent this, as if the download fails the transaction will be cancelled removing all changes.

Currently the user credentials are stored on the device unencrypted, inside the devices user preferences [42] for the application. This method of storing the credentials is secure under normal operation, but should a malicious user have root access to the device they will be able to extract this data from the user preferences. To secure this, the data needs to be encrypted before being stored on the device.

The key used for the encryption cannot be stored on the device, as that makes the encryption useless, as the hacker will be able to access the key and then be able to decrypt the password.

There are two methods of securing the key that I considered. One method is to store the key on a server, and then retrieve the key when the key is needed. Another method is to generate the key from a user entered value, such as a PIN, the user can then be asked to enter the PIN whenever the password needs to be decrypted.

5.8 Future improvements for the application

If extra time were available for this project, I would have added extra features and functionality. This section will outline any future improvements that could be added to the system.

5.8.1 Cross platform support

If extra time were provided, making native applications for the other top mobile operating systems would greatly improve the possible reach of the application. Although the models and classes have been written in Java, the design of the classes can be used to implement other operating systems.

5.8.2 Allow the user to add extra notes to drugs

Although not specifically stated by the NHS, I believe the staff would find the application more useful if they could add notes to an individual drug, these notes would be stored local to the device and would be shown alongside the current drug information. Due to the current database structure this could be implemented easily.

5.8.3 Allow the user to add calculator information

Currently the dataset downloaded for calculations from the NHS only contains around 20 drug calculator informations. Given extra time I would have allowed users to manually enter drug calculator information, thus allowing them to extend their local database.

This idea could be extended further, and the user could be able to submit there information to the live server, so the information is shared with all users. Due to the nature of the calculations, they would need to be verified by an admin before approval.

5.8.4 Partial database update

To update the database on the current system, the application first deletes all data within the database and then downloads the new data. This was the only method of implementing the update, due to the limitations of the provided API. Given more time I could've worked with the NHS to improve the API to allow for partial updates to the database, which would greatly improve the speed of an update.

5.8.5 Push notifications of database updates

Once partial updates had been implemented, it would be beneficial to the user, to get notified via push notifications when an update to the database is available. This would ensure that the users database is always up to date and would increase user engagement with the application.

5.8.6 Ability to reset credentials

Currently there is no method of resetting the users credentials, should they forget them. Provided extra time, I could have worked with the NHS to create API's for resetting the users credentials with the application.

5.8.7 Allow for online mode

Some users may not want to download the entire database before using the application. It would be nice to allow user to use the live database from the application, meaning a local copy of the database would not be stored and only informations that is needs is downloaded. Obviously this solution would not work when the user does not have Internet access, but could be useful in places where Internet is available, for example a doctor's surgery.

Appendices

Appendix A

Third-Party Code and Libraries

Android SDK Library The Android SDK Library [13] has been to create a native Android application. This library has provided the application with many useful functions, such as the user interface and database interactons. This

Robospice Library The Robospice Library [39] was used to create asynchronous download services. These services allowed the data download task to run in the background. Robospice is licenced under the Apache 2.0 licence [15]. This library was used without modification, although one class was duplicated and used within my project to add extra functionalilty.

Junit The JUnit framework [31] was used to execute unit tests on the completed implementation.

Appendix B

Code samples

2.1 Default request notifier

The default request notifier class was taken from the Robospice source code and then modified to notify the DataProgress singleton of a request completion.

```
 package com.octo.android.robospice.request.notifier;

import java.util.Set;

import roboguice.util.temp.Ln;
import android.os.Handler;
import android.os.Looper;
import android.os.SystemClock;

import com.octo.android.robospice.exception.
    RequestCancelledException;
import com.octo.android.robospice.persistence.exception.
    SpiceException;
import com.octo.android.robospice.request.CachedSpiceRequest;
import com.octo.android.robospice.request.listener.
    PendingRequestListener;
import com.octo.android.robospice.request.listener.
    RequestListener;
import com.octo.android.robospice.request.listener.
    RequestProgress;
import com.octo.android.robospice.request.listener.
    RequestProgressListener;

/**
 * Default implementation of RequestListenerNotifier. It will
     notify listeners
 * on the ui thread.
 * @author Andrew Clark
 */
public class DefaultRequestListenerNotifier implements
   RequestListenerNotifier {
    // ATTRIBUTES
    private final Handler handlerResponse;

    // CONSTRUCTOR     public DefaultRequestListenerNotifier() {
        handlerResponse = new Handler(Looper.getMainLooper());
    }

    private void post(final Runnable r, final Object token) {
        handlerResponse.postAtTime(r, token, SystemClock.
           uptimeMillis());
    }

    @Override
    public <T> void notifyListenersOfRequestNotFound(final
       CachedSpiceRequest<T> request, final Set<RequestListener
       <?>> listRequestListener) {
        post(new NotFoundRunnable(listRequestListener), request.
           getRequestCacheKey());
    }

    @Override
    public <T> void notifyListenersOfRequestAdded(final
       CachedSpiceRequest<T> request, Set<RequestListener<?>>
       listeners) {
        // does nothing for now
    }

    @Override
    public <T> void notifyListenersOfRequestAggregated(final
       CachedSpiceRequest<T> request, Set<RequestListener<?>>
       listeners) {
        // does nothing for now
    }

    @Override
    public <T> void notifyListenersOfRequestProgress(final
       CachedSpiceRequest<T> request, final Set<RequestListener
       <?>> listeners, final RequestProgress progress) {

        post(new ProgressRunnable(listeners, progress), request.
           getRequestCacheKey());
    }

    @Override
```

```
public <T> void notifyListenersOfRequestSuccess(final
   CachedSpiceRequest<T> request, final T result, final Set<
   RequestListener<?>> listeners) {

    post(new ResultRunnable<T>(listeners, result), request.
       getRequestCacheKey());
}

@Override
public <T> void notifyListenersOfRequestFailure(final
   CachedSpiceRequest<T> request, final SpiceException e,
   final Set<RequestListener<?>> listeners) {

    post(new ResultRunnable<T>(listeners, e), request.
       getRequestCacheKey());
}

@Override
public <T> void notifyListenersOfRequestCancellation(final
   CachedSpiceRequest<T> request, final Set<RequestListener
   <?>> listeners) {

    post(new ResultRunnable<T>(listeners, new
       RequestCancelledException("Request␣has␣been␣cancelled
       ␣explicitely.")), request.getRequestCacheKey());
}

@Override
public <T> void clearNotificationsForRequest(final
   CachedSpiceRequest<T> request, final Set<RequestListener
   <?>> listeners) {

    handlerResponse.removeCallbacksAndMessages(request.
       getRequestCacheKey());
}
// INNER CLASSES
private static class NotFoundRunnable implements Runnable {
    private final Set<RequestListener<?>> listeners;

    public NotFoundRunnable(final Set<RequestListener<?>>
       listeners) {
        this.listeners = listeners;
    }

    @Override
    public void run() {

        if (listeners == null) {
                return;
            }

            Ln.v("Notifying␣" + listeners.size() + "␣listeners␣
                of␣request␣not␣found");
            synchronized (listeners) {
                for (final RequestListener<?> listener :
                    listeners) {
                    if (listener != null && listener instanceof
                        PendingRequestListener) {
                        Ln.v("Notifying␣%s", listener.getClass()
                            .getSimpleName());
                        ((PendingRequestListener<?>) listener).
                            onRequestNotFound();
                    }
                }
            }
        }
    }

    private static class ProgressRunnable implements Runnable {
        private final RequestProgress progress;
        private final Set<RequestListener<?>> listeners;

        public ProgressRunnable(final Set<RequestListener<?>>
            listeners, final RequestProgress progress) {
            this.progress = progress;
            this.listeners = listeners;
        }

        @Override
        public void run() {

            if (listeners == null) {
                return;
            }

            Ln.v("Notifying␣" + listeners.size() + "␣listeners␣
                of␣progress␣" + progress);
            synchronized (listeners) {
                for (final RequestListener<?> listener :
                    listeners) {
                    if (listener != null && listener instanceof
                        RequestProgressListener) {
                        Ln.v("Notifying␣%s", listener.getClass()
                            .getSimpleName());
                        ((RequestProgressListener) listener).
                            onRequestProgressUpdate(progress);
                    }
                }
            }
        }
    }

    private static class ResultRunnable<T> implements Runnable {

        private SpiceException spiceException;
        private T result;
        private final Set<RequestListener<?>> listeners;

        public ResultRunnable(final Set<RequestListener<?>>
            listeners, final T result) {
            this.result = result;
            this.listeners = listeners;
        }

        public ResultRunnable(final Set<RequestListener<?>>
            listeners, final SpiceException spiceException) {
            this.spiceException = spiceException;
            this.listeners = listeners;
        }

        @Override
        public void run() {
            if (listeners == null) {
                return;
            }

            final String resultMsg = spiceException == null ? "
                success" : "failure";
            Ln.v("Notifying␣" + listeners.size() + "␣listeners␣
                of␣request␣" + resultMsg);
            synchronized (listeners) {
                for (final RequestListener<?> listener :
                    listeners) {
                    if (listener != null) {
                        @SuppressWarnings("unchecked")
                        final RequestListener<T> listenerOfT = (
                            RequestListener<T>) listener;
                        Ln.v("Notifying␣%s", listener.getClass()
                            .getSimpleName());
                        if (spiceException == null) {
                            listenerOfT.onRequestSuccess(result)
                                ;
                        } else {
                            listener.onRequestFailure(
                               spiceException);
                        }
                    }
                }
            }
        }
    }
}
```

Appendix C

Further documentation

This section includes all of the documents that were created whilst executing this project. Most of these documents were required by the waterfall model [3].

NHS Wales Informatics Service,
Brunel House,
2 Fitzalan Road,
Cardiff,
CF24 0HA
2nd May, 2014

Feedback on Aidan Fewster, student at Aberystwyth University

For the first time this year, NHS Wales Informatics Service became involved with Aberystwyth University when we proposed the development of a mobile phone app for the NHS Injectable Medicines Guide (IMG) as a possible dissertation project. The IMG is a website that that provides information on how to prepare and administer injectable drugs and is used by over 90% of NHS Trusts across the UK by nurses, doctors and pharmacists. It is a major contribution to patient safety and is recommended by the National Patient Safety Agency. We have been under pressure to develop a mobile phone app version of the website.

We supplied an outline design of the app. Aidan was one of two students to select this proposal to take forward as his dissertation. I am providing this feedback as a result of working with Aidan while he has written the software.

My contact with Aidan has been via email and he has always responded promptly to my questions and provided full and accurate answers. Aidan has had to ask us to provide access to the various data sources and I suspect the access that we have provided has not been what he would have chosen but he has always been willing to adapt to what we have provided and work with the our timescales. He has only requested minimal information so he must have worked within his own initiative to complete his work.

And then he delivered the app for us to look at. I was impressed that he had considered how we would implement the software and provided a simple means for us to load the app. Initially we had problems loading the software but Aidan researched the problem and worked with us to resolve the issue which was largely due to the age of our equipment. Implementation and support is a different skill set to development and I was impressed by his problem-solving skills and his persistence in resolving the issue.

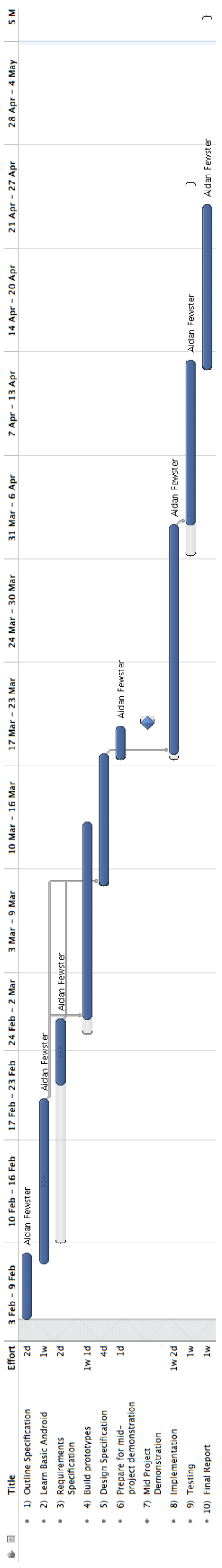
I was bowled over by the app that he has produced; it is well beyond our outline design. He has clearly had to adapt what was essentially an iPhone design to the Android platform and he has produced a design that is very clean and easy to use. He has clearly understood the user interface design principles and I much prefer his design of the pages and the navigation from one function to another over our design.

One aspect of the app is the ability to download and update the data from the central website and to manage the security of this function. Aidan has demonstrated that he understands the security issue and has devised an update that is simple to use but downloads the complex data sets that are involved and has been able to transform the datasets into the information as it appears on the screen, matching across the keys to the each file.

I have demonstrated the app to the IMG Executive Group which consists of Pharmacists from Imperial NHS Trust and an RCN nurse expert on injectable drugs and they were very impressed and found the app intuitive and easy to use. The view was that nurses, for whom very few apps exist and so are not expert users, would find welcome this app and find it easy to navigate.

Overall we are very impressed with the way that Aidan has interacted with us and with the product that he has produced. I am sure he will have an excellent career in software if this is the path that he chooses.

Robin Burfield, Development Manager, NHS Wales Informatics Service.



Title	Effort
1) Outline Specification	2d
2) Learn Basic Android	1w
3) Requirements Specification	2d
4) Build prototypes	1w 1d
5) Design Specification	4d
6) Prepare for mid-project demonstration	1d
7) Mid Project Demonstration	
8) Implementation	1w 2d
9) Testing	1w
10) Final Report	1w

NHS Wales formulary and antimicrobial Android application

Report Name	Outline Project Specification
Author (User Id)	Aidan Wynne Fewster (awf1)
Supervisor (User Id)	Andrew Starr (aos)
Module	CS39440
Degree Scheme	G400 (Computer Science)
Date	February 10, 2014
Revision	1.0
Status	Release

1 Project description

NHS Wales have requested a mobile application to aid NHS staff in administering a variety drugs and medicines. The NHS own a database which contains a list available drugs along with their usage, dosage and other useful information. They would like a mobile application to access this database and provide the information to their staff.

The data contained within the database is updated whenever a new drug is introduced, a drugs information changes or a drug is removed from the database. An internet connection may not be available throughout an NHS establishment therefore the database must be stored on the device for offline use. As the information is used to administer drugs to patients it is vital that the application provides the most up-to date and accurate information. Due to this the NHS database and the applications database must be accurately synchronised whenever possible.

To access the application and it's data a member of NHS staff must first provide correct login credentials in order to authenticate themselves. Should a user forget their login credentials they must be able to reset their password. Upon authentication the user will be able to view recently updated drugs and search for available drugs using a search field (with intelligent suggestions). Once a drug has been found, the user will be neatly presented with information about that drug. The user will also be able to enter a patients weight which will then be provide the user with dosage requirements for that patient. As the patients weight will be entered manually by the user, this field must be heavily validated.

The NHS have raised concerns for the security of their database as releasing the database to the public could cause problems for the NHS. Therefore it is important that the data and all communications of the data are suitably encrypted. It is also important that the application is not released to the public via the application store or any other means. I will only be provided with a subset of the database from the NHS for security reasons.

As the application will be used to administer potentially lethal drugs, a thorough testing strategy will need to be executed throughout. Unit testing will need to be carried out extensively on the formula for calculating dosage requirements. Testing will also be used to ensure that the correct data is displayed to the user.

As to improve the maintainability and customisability of the system the NHS have asked that the structure of the database to be outlined within an XML file (or other text editable file), this will allow them to create multiple applications for a variety databases using the same application code.

2 Proposed tasks

I will be creating the application as an Android application to be used on Android devices running a version of Android above 4.0 (ice cream sandwich). I will adhere to the design rules specified by the Android design document [1] in order to create an application that feels native to the Android experience.

I propose to retrieve database updates from the database using a JSON [6] API which will be transmitted over SSL after user authentication for security reasons. I also propose for user authentication and resetting of user credentials to be achieved using the JSON API. I will need to research how to interact with JSON API's within Android in order to achieve this. I have yet to have a meeting with the NHS representative yet, therefore I do not know if a JSON API will be available to me.

Once the user has authenticated themselves the database on the device will be synchronised with the NHS database, I need to research the best methods of synchronisation. The user will then be able to search through the database by either scrolling through a list view or typing part of the drugs information in the search box (The application will provide intelligent suggestions from partially entered information). I need to figure out the best method of generating suggestions, one method is using content providers [3].

I propose to provide the user with a notification [2] when an update to a drug has been made. I will need to learn how to send notifications to an Android device.

I would like to encrypt the database stored on the device in order to prevent unauthorised personnel accessing the data. One method of achieving this is by using SQLCipher [4], I would need to learn how to integrate SQLCipher into an Android application to achieve this.

As mentioned previously the application must be tested thoroughly in order to ensure that the correct information is outputted. I will need to learn how to efficiently test using the android test framework [5] and integrate these tests into the project.

3 Project deliverables

Requirements specification This document will list all the requirements for the systems and outline the features of the final system. This document can later be used to test that the system meets the required needs.

Test specification A comprehensive test specification must also be provide so that the NHS can see that the application has been thoroughly tested as well as guide them in executing the tests for themselves. This will improve the NHS's confidence within the final application.

Final Android Application I should provide a usable, stable and secure Android application that will achieve the tasks outlined within the requirements specification. The application should be aesthetically pleasing as well as be intuitive for the user to use. The application should be packaged ready for distribution on a multitude of devices.

Documentation This project will also include a large amount of documentation, this is has even greater importance with this project as this project will be delivered to a large organisation, therefore the documentation must be comprehensive in order to improve maintainability of the provided system. Documentation will include design decisions made and documents to support the design (UML). The documentation will also provide a list of libraries that have been used, their licences and the reasons the library has been used.

User Manual As the system will be used by staff of the NHS, who may not be technically minded, I will provide a user manual for the users so that they can learn how to use the application correctly and effectively.

Final Report This is the full report that will include all the documentation created. It will outline design choices that have been made, any changes from the original requirement specifications, any issues I have found whilst executing this project, my diary entries I have made whilst working on the project, acceptance testing with the NHS and a self evaluation.

Annotated Bibliography

[1] Google, “Android design,” https://developer.android.com/design/index.html, accessed February 2014.

This document is useful as it provides information on how to design android application so that they feel intuative to the user given they have a knowledge of how to use Android.

[2] ——, “Android notifications,” http://developer.android.com/guide/topics/ui/notifiers/notifications.html, accessed February 2014.

This document provides information on how to send notifications to an Android device from your application.

[3] ——, “Content providers,” http://developer.android.com/guide/topics/providers/content-providers.html, accessed February 2014.

This document is useful as it provides information on content providers for Android. Content providers allow you to distribute your database between multiple applications on your device as well as provide help with intelligent suggestions.

[4] SQLCipher, “Sqlcipher homepage,” http://sqlcipher.net/, accessed February 2014.

This webpage provides the download and documentation for the SQLCipher library. SQLCipher allows you to easily encrypt and SQLite database.

[5] L. Vogel, “Android application testing with the android test framework,” http://www.vogella.com/tutorials/AndroidTesting/article.html, Sept. 2011, accessed February 2014.

This document provides information on how to effectively test Android applications using the Android test framework.

[6] W3C, “Json specification,” http://www.w3.org/TR/2014/REC-json-ld-20140116/, accessed February 2014.

This document provides the specification and structure of JSON.

NHS Wales formulary and antimicrobial Android application

Report Name	Requirements Specification
Author (User Id)	Aidan Wynne Fewster (awf1)
Supervisor (User Id)	Andrew Starr (aos)
Module	CS39440
Degree Scheme	G400 (Computer Science)
Date	May 2, 2014
Revision	1.1
Status	Release

1 Introduction

1.1 Purpose of this document

This document will outline the functional, user interface and security requirements for the NHS Wales formulary and antimicrobial Android application. It will also include use cases for the system. This document can later be used during testing to ensure that the system meets all the requirements outlined within.

1.2 Project overview

The main focus of this project is to produce a mobile application for Android which will aid NHS medical staff in obtaining useful information for injectable medicines which are available within the NHS . For more information refer to the Outline Specification document.

2 Project details

This section will describe the general factors that affect the requirements. It will describe the characteristics of the users using the system and explain how they shall be using the system. I will also outline any assumptions that I have made.

2.1 User characteristics

The primary users of the application will be medical staff of NHS Wales. They will be using the applications to search for information on injectable medicines that are available within the NHS. They should already have broad knowledge of the information displayed within the application.

As the application will be built on the Android platform the member of staff should already be familiar with the user interface of Android and therefore be able to natively use applications that follow the Android design guidelines.

2.2 Assumptions

As the application will be used in hospitals I am assuming that an internet connection will not be available at all times when the application is in use, therefore the data used by the application must be stored for offline use.

Another assumption I am making is that the member of staff will be installing the application on a device they are familiar with and therefore know how to use the basics of the Android operating system.

2.3 Dependencies

The application must have access to the NHS database in order to download the data required for the application. This access is provided via an XML API with a basic HTTP GET authentication procress (username and password are sent within the URL). All data transfered to and from the API is sent over HTTP without encryption (personally I would use an encypted connection for security reasons, but when asking the NHS about the secuirty of the data they said that secuirty was not a concern of theirs).

3 Requirements

3.1 Functional requirements

This section will list the functional requirement's (FR)'s for this project.

FR 1	**Authentication**
FR 1.1	User must be able to authenticate themselves using their credentials
FR 1.2	User must be notified if the password they enter is incorrect
FR 1.3	User will be notified if the authentication failed due to connection issues
FR 1.4	User must be able to logout of the system, removing all data
FR 2	**Database synchronisation**
FR 2.1	After login or when the user presses update the application must truncate all database tables and begin downloading new data
FR 2.2	Download complete list of drug indexes from database
FR 2.3	Download complete list of drugs and drug informations
FR 2.3	Download all information needed for calculating doses and infusion rates.
FR 2.4	Download must still run when the application is in the background
FR 3	**Menu options**
FR 3.1	Upon pressing the Menu button on the device the user will be presented with a list of available options, which execute tasks (Logout, exit, search)
FR 4	**Main screen**
FR 4.1	Upon successful data download the user will be displayed with a screen where they can navigate to other parts of the application
FR 4.2	User will be see when an update was last performed, and perform an update from this screen.
FR 5	**Browse drugs**
FR 5.1	This screen will allow the user to view a list of all drugs
FR 5.2	There will be an input box on this screen, when the user enters text into the input box the results in the list will be filtered to only show results related to the input
FR 5.3	The user will be able to click a drug in the list to open a new screen displaying the needed information
FR 6	**View drug**
FR 6.1	When a drug has been selected the drug and all its information will be displayed in an easy to read format
FR 6.2	Where drug information headers contain help information, a help icon will be displayed next to the header.
FR 6.3	When heading help icon is clicked the helping information will be displayed
FR 6.4	If the drug had calculator information, then a button to open the calculator should be shown

FR 7	**Browse drugs with calculators**
FR 7.1	A view similar to the browse drugs view will allow the browsing of drugs that contain calculator information.
FR 8	**Calculate dose and infusion rate**
FR 8.1	The user will be able to select calculation type
FR 8.2	User will be able to enter information required for the calculation
FR 8.3	When the calculate button has been clicked the input will be thoroughly validated
FR 8.4	After validation the result of the calculation will be displayed to the user
FR 8.5	The equation and values used to calculate the answer will be neatly displayed to the user
FR 9	**XML customisability for developers**
FR 9.1	All text within the application must be changeable through XML files.
FR 9.2	The structure of the XML APIs provided must be outline within XML files, allowing easy customisation for different APIs

3.2 User interface requirements

The user interface for the application should follow the guidelines set out in the Android developer design guide. Following these guidelines will allow the user to learn to use the application with minimalistic effort.

When the user first opens the application, they will be presented with a login field. Upon logging in they will be presented with the latest updates to drugs and buttons. The buttons will allow the user to begin searching for a drug, force an update of the database and open the about page.

When the user selects a drug they will be neatly presented with a page containing sections of information. This page will also contain a button to open the dosage calculation.

The user interface must be simplistic but make it clear what action you are currently performing. For example when calculating the dosage of penicillin for a patient, the user must be able to clearly see that penicillin has been selected and the weight of the patient used to calculate the dosage.

4 Use case diagram

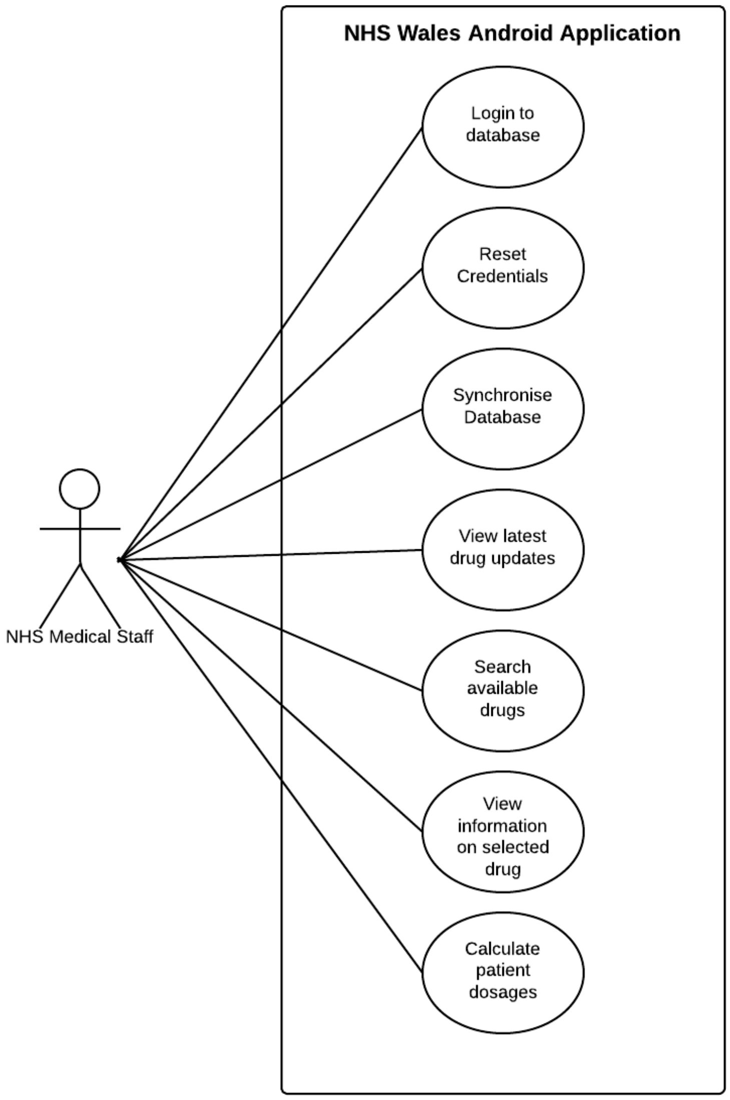


Figure 1: Use case diagram for the NHS application

4.1 Use case description

On the first time opening the application the user will be requested to login. Upon logging in the application will download a complete set of the data for offline use. After the download process has complete the user will be be displayed with a welcome screen, allowing them to navigate to the search page.

On the search page the user will be able to scroll through the list of drugs that have been downloaded or enter part of the drug name they're searching for in the search box to limit the result to appropiate matches.

Once the user has found the drug they're searching for they can select it to be displayed with all the information available for that drug.

If the drug selected has calculator information provided (from the database) the user will be presented with a button to calculate dose or infusion rate.

Upon clicking the calculator button the user will be presented with input fields to enter values needed to caluclate dose or infusion rate. When the user clicks on the calculate button they wlll be displayed with the answer to calculation and shown the equations used to reach that answer.

Whilst the application is running, the user will be able to press the menu button and be presented with a menu of options to complete tasks such as update, logout, exit.

NHS Wales injectable medicines guide Android application

Report Name	Design Specification
Author (User Id)	Aidan Wynne Fewster (awf1)
Supervisor (User Id)	Andrew Starr (aos)
Module	CS39440
Degree Scheme	G400 (Computer Science)
Date	March 20, 2014
Revision	0.8
Status	Draft

1 Introduction

1.1 Purpose of this document

The purpose of the document is to provide documentation for the architectural design of the NHS Wales injectable medicines guide Android application. It will show the class's which should be created and the methods within them, as well as how they're linked, this will be achieved through a class diagram. This document will also outline the project user interface design through UI mock-up wireframes.

1.2 Project overview

The main focus of this project is to produce a application for Android which will aid NHS medical staff in obtaining information on injectable medicines. The user will be able to search through a list of drugs by there title. Upon selecting a drug they will be able to view a detailed monograph for that drug. User will also be able to calculate dosage and infusion rates using the application.

2 Architectural design

This section will contain the architectural for the mobile application. This section will include a use case diagram and a class diagram for the system.

2.1 Use case diagram

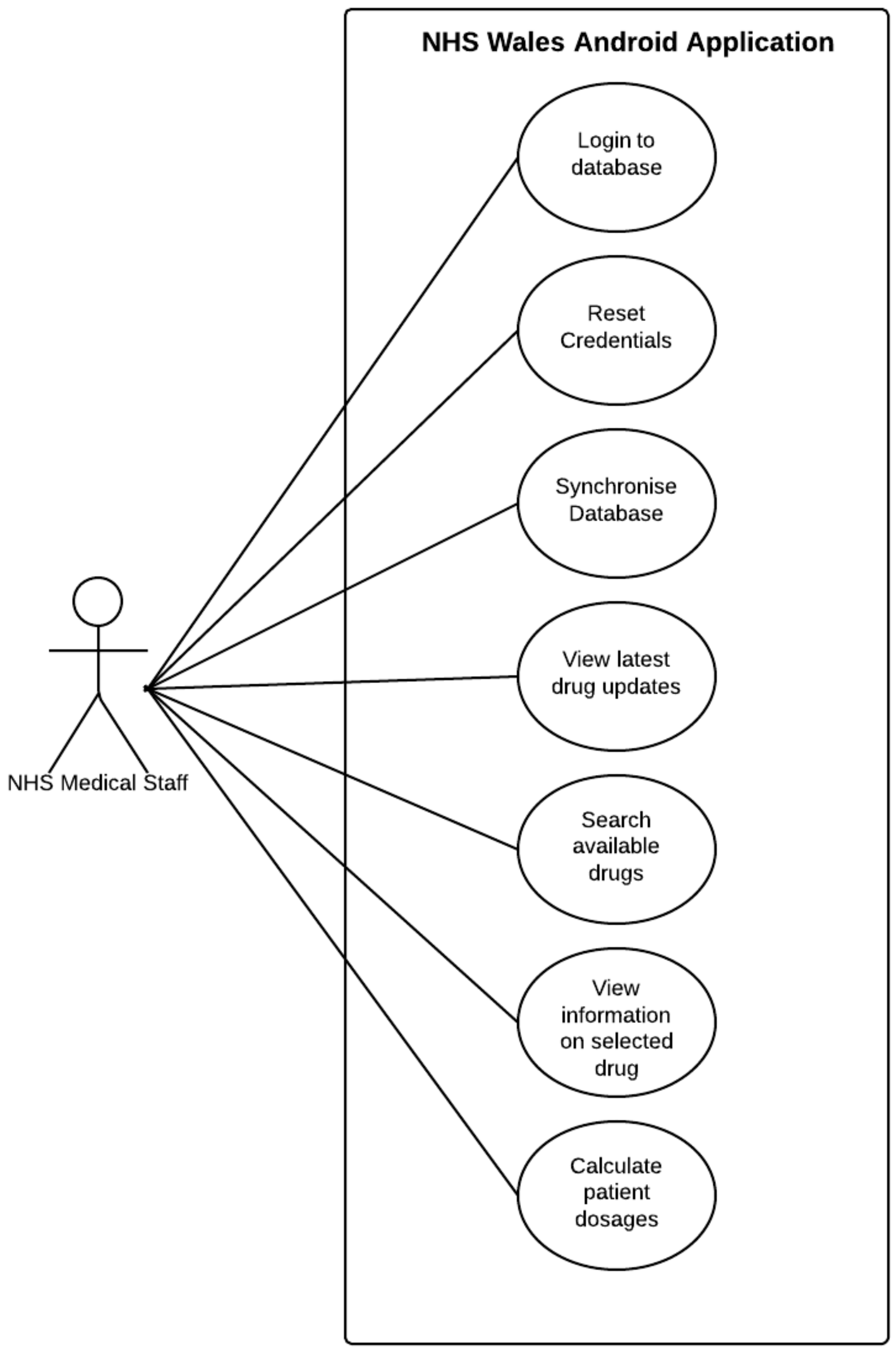


Figure 1: Use case diagram for the NHS application

2.1.1 Use case descriptions

Login to database A user will be able to authenticate themselves with the database using their NHS login credentials.

Synchronise database Upon authenticating themselves the user will be able to download a complete copy of the available database to their device.

View latest drug updates The user will how many monographs have been added or removed from the live database since they last synchronised their local one.

Search available drugs The user will be able to easily search through all the drugs within the database. The search will automatically suggest suitable results.

View information on selected drug Upon selecting a drug the user will be able to view a monograph for that drug. They will be able to select each section of the monograph they would like to read.

Calculate patient dosages The user will be able to calculate the infusion rate and dosage for a given patient. This information will be validated for user input error.

2.2 Class diagram

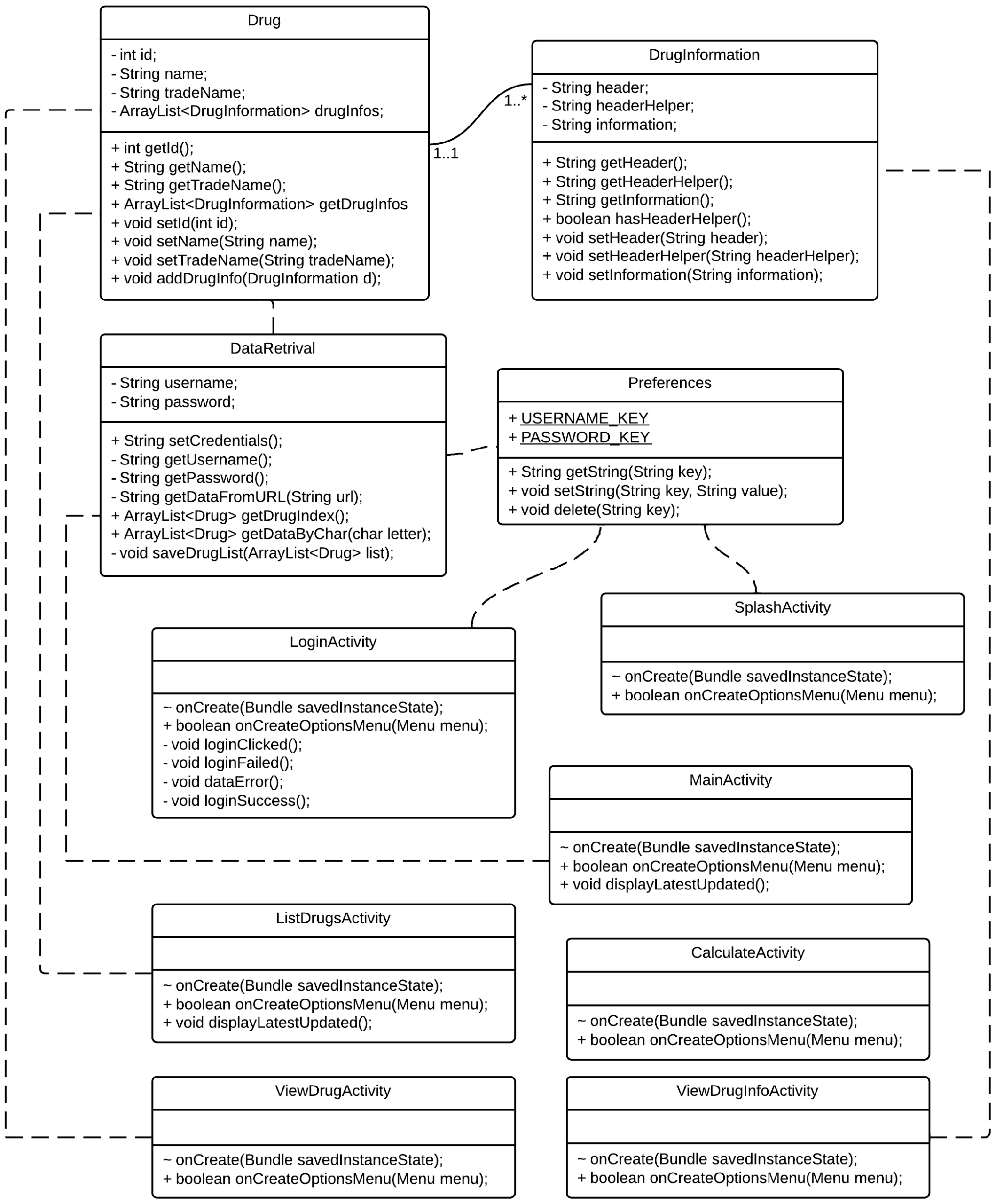

Drug
- int id;
- String name;
- String tradeName;
- ArrayList<DrugInformation> drugInfos;
+ int getId();
+ String getName();
+ String getTradeName();
+ ArrayList<DrugInformation> getDrugInfos
+ void setId(int id);
+ void setName(String name);
+ void setTradeName(String tradeName);
+ void addDrugInfo(DrugInformation d);
1..1
1..*
DrugInformation
- String header;
- String headerHelper;
- String information;
+ String getHeader();
+ String getHeaderHelper();
+ String getInformation();
+ boolean hasHeaderHelper();
+ void setHeader(String header);
+ void setHeaderHelper(String headerHelper);
+ void setInformation(String information);
DataRetrival
- String username;
- String password;
+ String setCredentials();
- String getUsername();
- String getPassword();
- String getDataFromURL(String url);
+ ArrayList<Drug> getDrugIndex();
+ ArrayList<Drug> getDataByChar(char letter);
- void saveDrugList(ArrayList<Drug> list);
Preferences
+ USERNAME_KEY
+ PASSWORD_KEY
+ String getString(String key);
+ void setString(String key, String value);
+ void delete(String key);
SplashActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
LoginActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
- void loginClicked();
- void loginFailed();
- void dataError();
- void loginSuccess();
MainActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
+ void displayLatestUpdated();
ListDrugsActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
+ void displayLatestUpdated();
CalculateActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
ViewDrugActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
ViewDrugInfoActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);

Figure 2: Use case diagram for the NHS application

2.2.1 Class diagram descriptions

Drug This class represents a drug in the database

DrugInformation This class represents a piece of the monographs information about the drug.

DataRetrival This class is responsible for authenticating the user and synchronising the local database with the live database.

Preferences This class is responsible for accessing the SharedPreferences of the application. It will be used to store and retrieve preferences

SlashActivity This is the loading screen when the app first launches

LoginActivity This is the activity which the user will use to authenticate themselves

MainActivity This class for the main activity, this activity will alert the user of any changes within the live database as well as let the user update their database and begin searching.

ListDrugsActivity Class for the list drugs activity, this activity will list all drugs stored in the database, allowing users to search through them.

ViewDrugActivity Class for the view drug activity, this activity will display simple information about the drug and allow the user to navigate to detailed information

ViewDrugInfoActivity Class for the drug information activity, this activity will display detailed information on a specific piece of the drugs monograph.

CalculateActivity Class for the calculate activity, this activity will allow the user to calculate dosage and infusion rates for a patient.

NHS Wales Injectable Medicine Guide Android application

Report Name	Design Specification
Author (User Id)	Aidan Wynne Fewster (awf1)
Supervisor (User Id)	Andrew Starr (aos)
Module	CS39440
Degree Scheme	G400 (Computer Science)
Date	May 4, 2014
Revision	1.0
Status	Release

1 Introduction

1.1 Purpose of this document

The purpose of the document is to provide documentation for the architectural design of the NHS Wales injectable medicines guide Android application. It will show the class's which should be created and the methods within them, as well as how they're linked, this will be achieved through a class diagram. This document will also outline the project user interface design through UI mock-up wireframes.

1.2 Project overview

The main focus of this project is to produce a application for Android which will aid NHS medical staff in obtaining information on injectable medicines. The user will be able to search through a list of drugs by there title. Upon selecting a drug they will be able to view a detailed monograph for that drug. User will also be able to calculate dosage and infusion rates using the application.

2 Architectural design

The application will be split up into separate packages, each performing a specific task. These packages will be called named activities, data download, database and the models package.

Separating the classes in related modules improves the re-usability and maintainability of the code. As future developers will easily be able to find the task of each class from its module.

The following sections will state the overall use case and provide class diagrams for each module.

2.1 Use case diagram

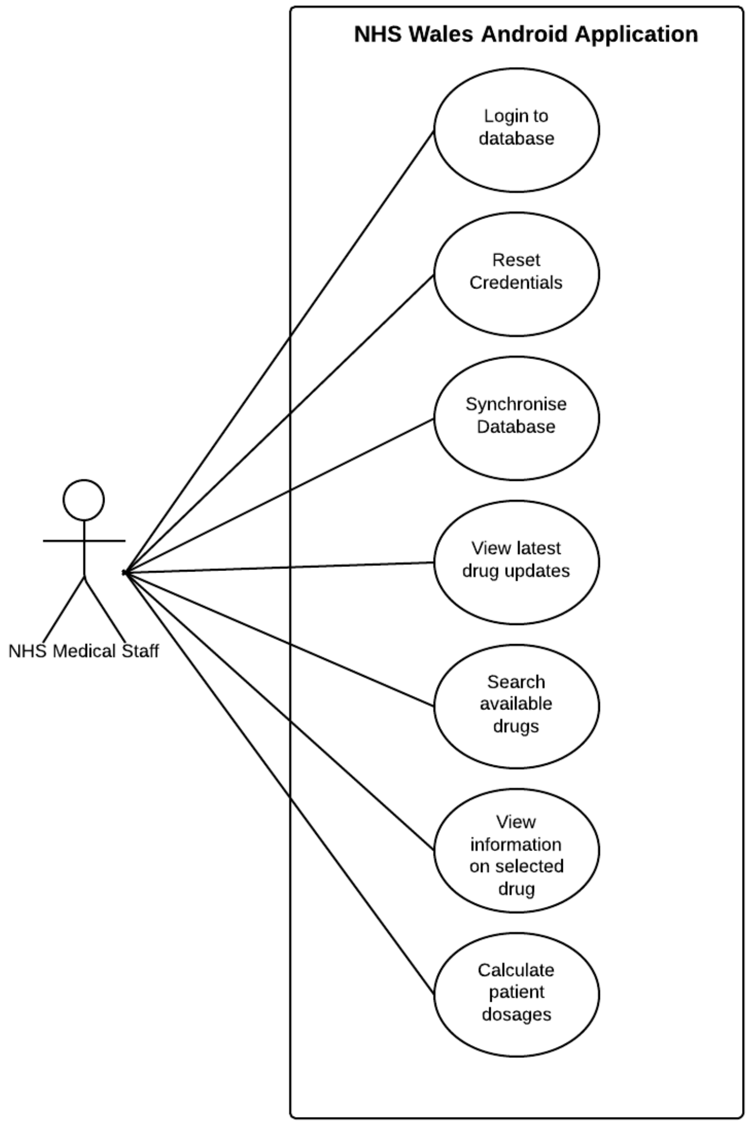


Figure 1: Use case diagram for the application

2.1.1 Use case descriptions

Login to database A user will be able to authenticate themselves with the database using their NHS login credentials.

Synchronise database Upon authenticating themselves the user will be able to download a complete copy of the available database to their device.

View latest drug updates The user will how many monographs have been added or removed from the live database since they last synchronised their local one.

Search available drugs The user will be able to easily search through all the drugs within the database. The search will automatically suggest suitable results.

View information on selected drug Upon selecting a drug the user will be able to view a monograph for that drug. They will be able to select each section of the monograph they would like to read.

Calculate patient dosages The user will be able to calculate the infusion rate and dosage for a given patient. This information will be validated for user input error.

2.2 Models package class diagram

The models package contains the model classes for each of the database tables. A model represents an item created from the database, for example a drug or a piece of information about a drug. Keeping all models within one package allows the models to be reused in other applications.

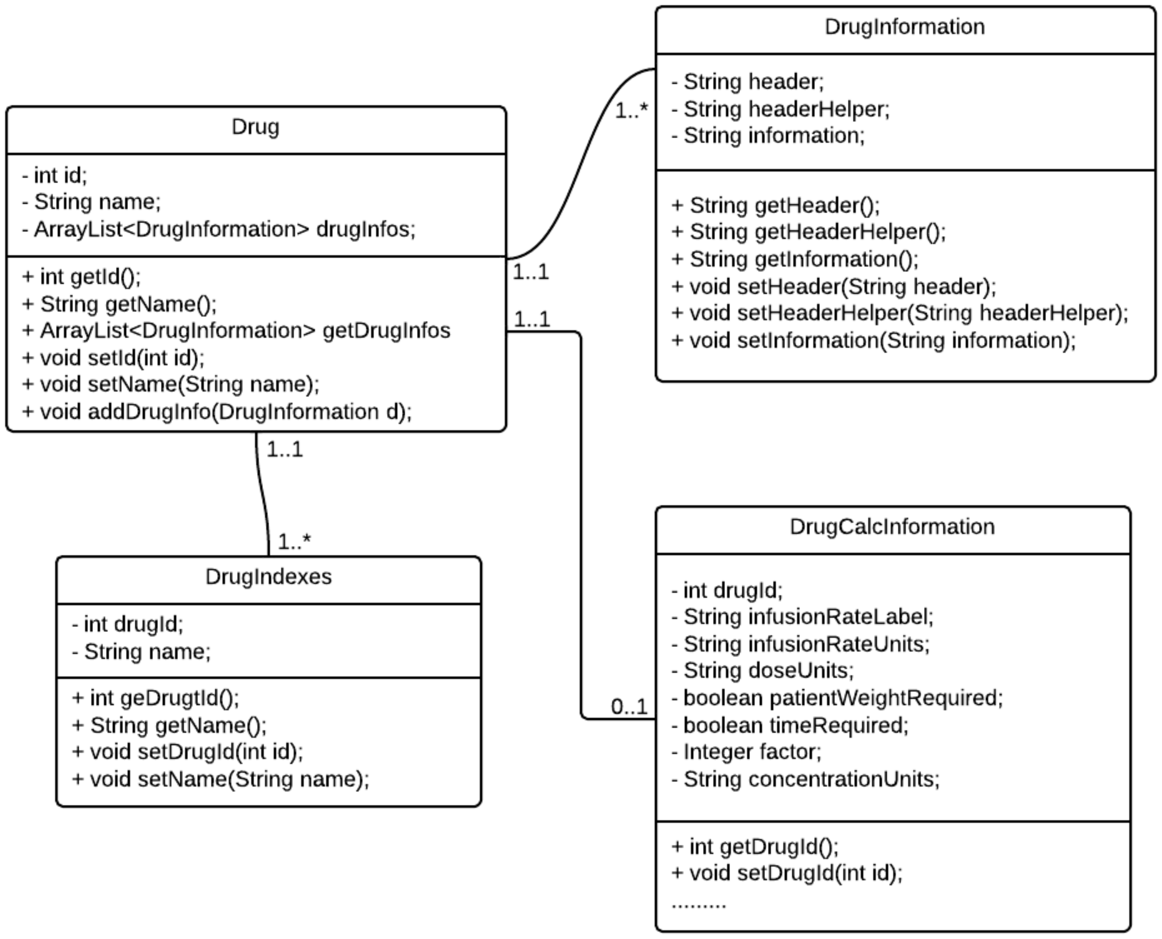

Drug
- int id;
- String name;
- ArrayList<DrugInformation> drugInfos;
+ int getId();
+ String getName();
+ ArrayList<DrugInformation> getDrugInfos
+ void setId(int id);
+ void setName(String name);
+ void addDrugInfo(DrugInformation d);
DrugInformation
- String header;
- String headerHelper;
- String information;
+ String getHeader();
+ String getHeaderHelper();
+ String getInformation();
+ void setHeader(String header);
+ void setHeaderHelper(String headerHelper);
+ void setInformation(String information);
DrugIndexes
- int drugId;
- String name;
+ int geDrugtId();
+ String getName();
+ void setDrugId(int id);
+ void setName(String name);
DrugCalcInformation
- int drugId;
- String infusionRateLabel;
- String infusionRateUnits;
- String doseUnits;
- boolean patientWeightRequired;
- boolean timeRequired;
- Integer factor;
- String concentrationUnits;
+ int getDrugId();
+ void setDrugId(int id);
....
1..*
1..1
1..1
0..1
1..1
1..*

Figure 2: Class diagram for the models package

2.2.1 Class diagram descriptions

Drug This class represents a drug in the database

DrugIndex This class represents a drug index, a drug index is a searchable name for a drug.

DrugInformation This class represents a piece of the monographs information about the drug.

DrugCalcInformation This class a set of information needed to perform a calculation

2.3 Database package class diagram

This package contains all classes that interact with the SQLite database. Currently this package only contains one class, but if the application expands, more classes can be added to split up the classes.

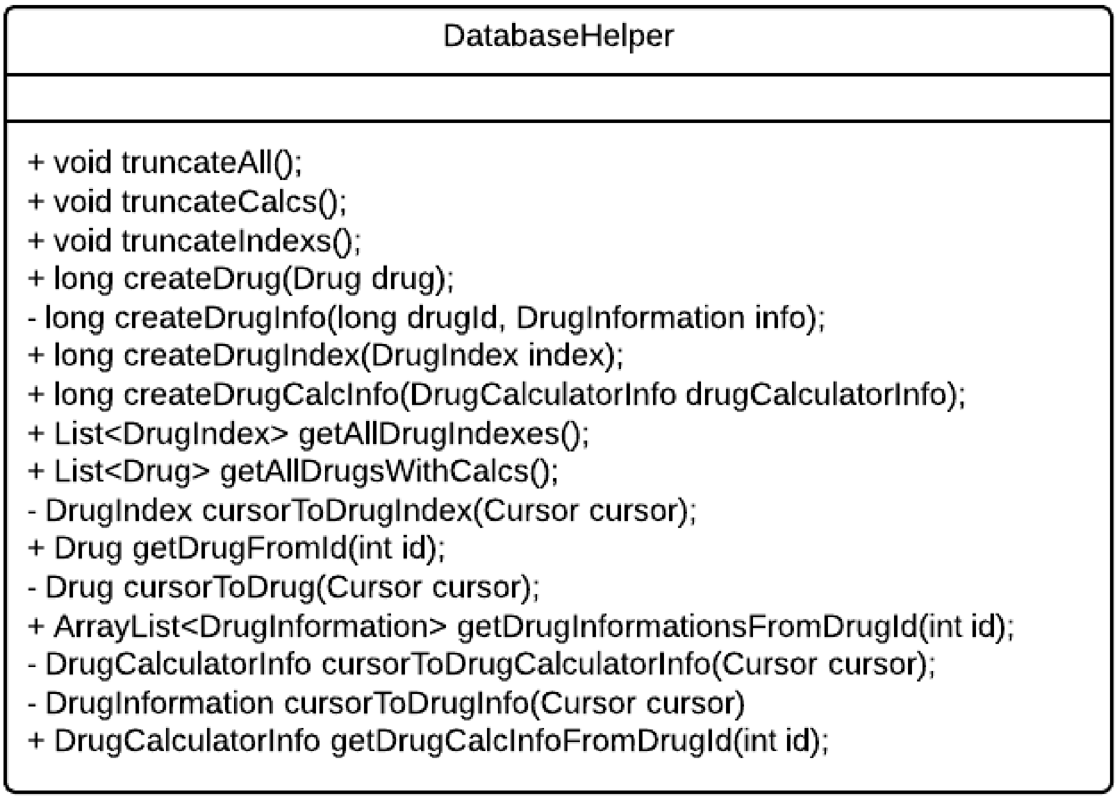

DatabaseHelper
+ void truncateAll();
+ void truncateCalcs();
+ void truncateIndexs();
+ long createDrug(Drug drug);
- long createDrugInfo(long drugId, DrugInformation info);
+ long createDrugIndex(DrugIndex index);
+ long createDrugCalcInfo(DrugCalculatorInfo drugCalculatorInfo);
+ List<DrugIndex> getAllDrugIndexes();
+ List<Drug> getAllDrugsWithCalcs();
- DrugIndex cursorToDrugIndex(Cursor cursor);
+ Drug getDrugFromId(int id);
- Drug cursorToDrug(Cursor cursor);
+ ArrayList<DrugInformation> getDrugInformationsFromDrugId(int id);
- DrugCalculatorInfo cursorToDrugCalculatorInfo(Cursor cursor);
- DrugInformation cursorToDrugInfo(Cursor cursor)
+ DrugCalculatorInfo getDrugCalcInfoFromDrugId(int id);

Figure 3: Class diagram for the database package

2.3.1 Class diagram descriptions

DatabaseHelper This class is responsible for fetching, creating and deleting data from the database. Having this class within its own package will allow the database and database code to be reused within other applications. If the code within the database helper class becomes to large using a package will also allow the code to be broken into smaller classes.

2.4 Activities package class diagram

Activities are Android's views. An activity provides a method of displaying information on screen for the user. The activities package contains the implemented classes for the activities.

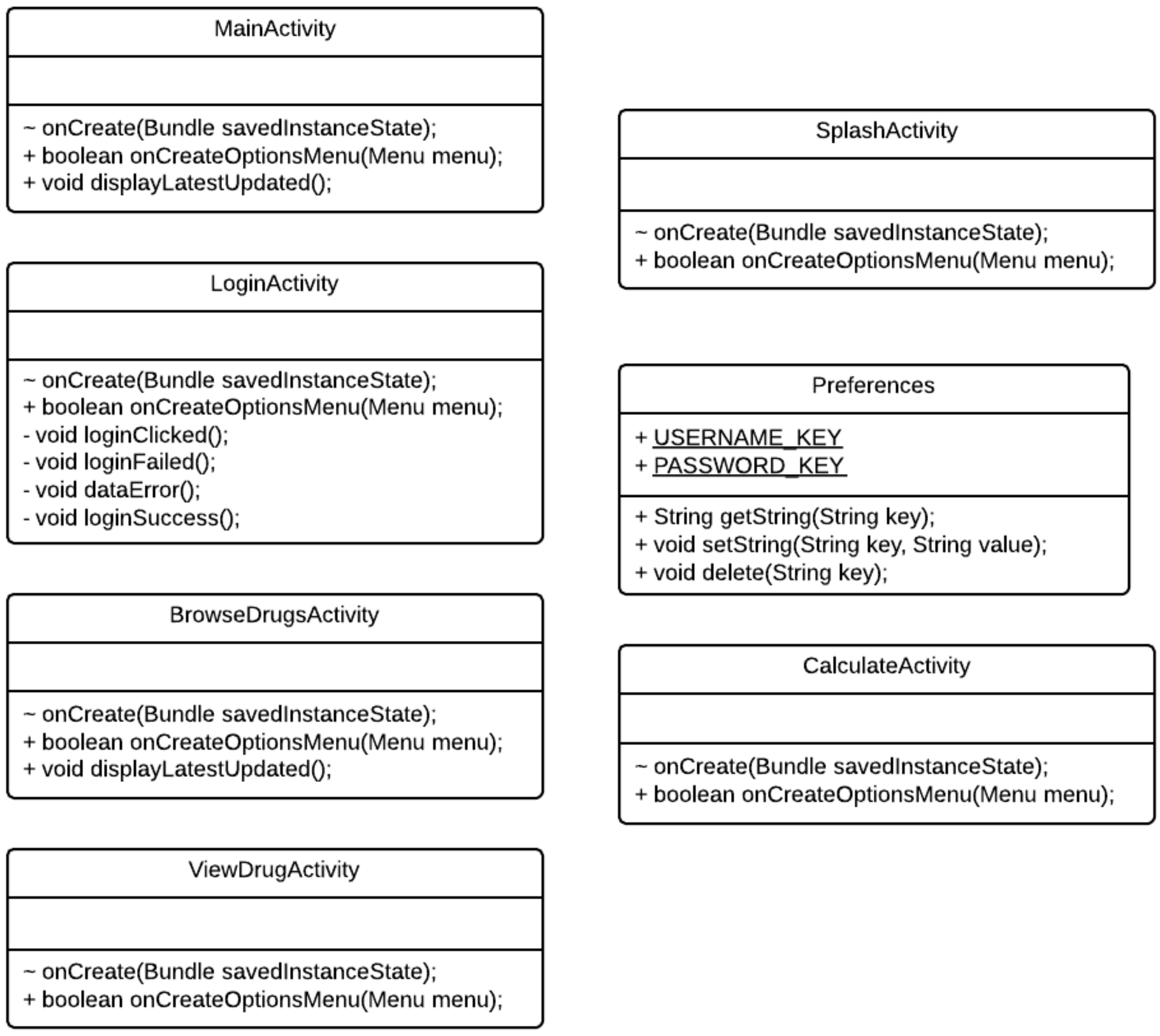

MainActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
+ void displayLatestUpdated();
LoginActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
- void loginClicked();
- void loginFailed();
- void dataError();
- void loginSuccess();
BrowseDrugsActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
+ void displayLatestUpdated();
ViewDrugActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
SplashActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);
Preferences
+ USERNAME_KEY
+ PASSWORD_KEY
+ String getString(String key);
+ void setString(String key, String value);
+ void delete(String key);
CalculateActivity
~ onCreate(Bundle savedInstanceState);
+ boolean onCreateOptionsMenu(Menu menu);

Figure 4: Class diagram for the activities package

2.4.1 Class diagram descriptions

MainActivity Shown once the user has successfully downloaded the database. This is the first screen a reoccurring user is sent to. This page allows the user to navigate to other screens.

LoginActivity This is the first screen the user sees when opening the application for the first time. Once they login their details are saved so they should only have to login once.

DownloadActivity This page is shown when the user is downloading the database for the first time, or when they are updating the database. The tasks within this screen can take several minutes so a progress bar helps to show to the user that the task is still running and hasnt crashed.

BrowseDrugsActivity These two screens are very similar; they both display a list of drugs and allow you to filter the list by entering part of the drugs name, inside the search box. The browse drugs page contains a list of all drugs and the browser calculators screen shows all drugs that have calculators.

ViewDrugActivity This screen is displayed when the user selects a drug from the browse drugs page. It contains the complete monograph for the drug, some headers contain an icon next to the header, if the user clicks a head with this icon, and helper information for that header is displayed within a dialog.

Calculator screen This is the screen where the user enters patient information and is displayed with the either the dose or infusion rate for the drug. When the result of the calculation is displayed, the equation used to perform the calculation is also displayed and described.

2.5 Data download package class diagram

The data download package contains the Robospice requests classes, which are responsible for downloading the individual API URLs and then parsing the data obtained. Each class within this package downloads a specific API XML file. Having all the data download classes in one package makes it easy to add new classes to download additional data in the future.

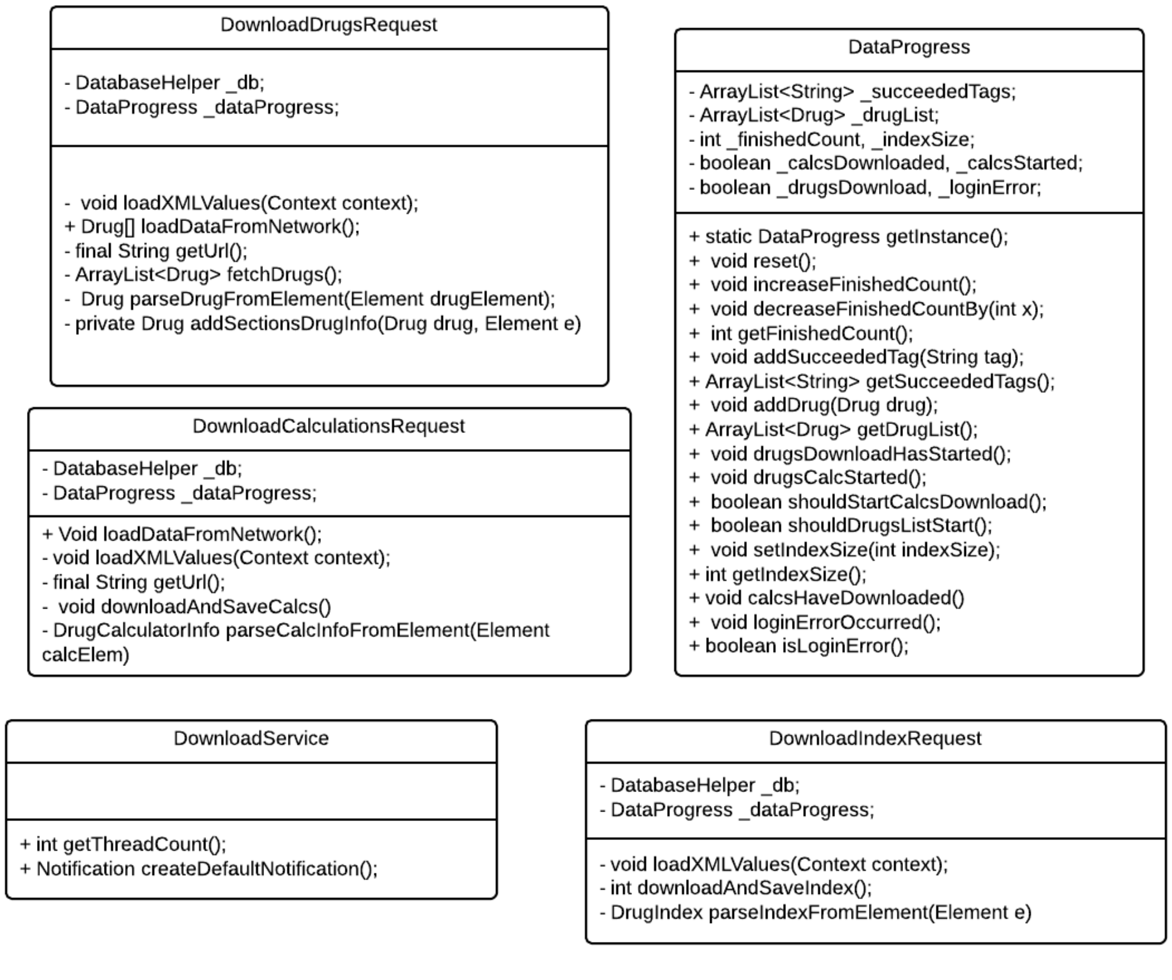

DownloadDrugsRequest
- DatabaseHelper _db;
- DataProgress _dataProgress;
- void loadXMLValues(Context context);
+ Drug[] loadDataFromNetwork();
- final String getUrl();
- ArrayList<Drug> fetchDrugs();
- Drug parseDrugFromElement(Element drugElement);
- private Drug addSectionsDrugInfo(Drug drug, Element e)
DownloadCalculationsRequest
- DatabaseHelper _db;
- DataProgress _dataProgress;
+ Void loadDataFromNetwork();
- void loadXMLValues(Context context);
- final String getUrl();
- void downloadAndSaveCalcs()
- DrugCalculatorInfo parseCalcInfoFromElement(Element calcElem)
DataProgress
- ArrayList<String> _succeededTags;
- ArrayList<Drug> _drugList;
- int _finishedCount, _indexSize;
- boolean _calcsDownloaded, _calcsStarted;
- boolean _drugsDownload, _loginError;
+ static DataProgress getInstance();
+ void reset();
+ void increaseFinishedCount();
+ void decreaseFinishedCountBy(int x);
+ int getFinishedCount();
+ void addSucceededTag(String tag);
+ ArrayList<String> getSucceededTags();
+ void addDrug(Drug drug);
+ ArrayList<Drug> getDrugList();
+ void drugsDownloadHasStarted();
+ void drugsCalcStarted();
+ boolean shouldStartCalcsDownload();
+ boolean shouldDrugsListStart();
+ void setIndexSize(int indexSize);
+ int getIndexSize();
+ void calcsHaveDownloaded()
+ void loginErrorOccurred();
+ boolean isLoginError();
DownloadService
+ int getThreadCount();
+ Notification createDefaultNotification();
DownloadIndexRequest
- DatabaseHelper _db;
- DataProgress _dataProgress;
- void loadXMLValues(Context context);
- int downloadAndSaveIndex();
- DrugIndex parseIndexFromElement(Element e)

Figure 5: Class diagram for the data download package

2.5.1 Class diagram descriptions

DataProgress This is class is used for checking the progress of requests and for displaying errors within the data download activity. This is a singleton class, so only one object of this class can be nstantiated.

DownloadCalculationsRequest This class is responsible for downloading the drug calculator informations

DownloadDrugsRequest This class is responsible for downloads the drugs and their informations

DownloadDrugIndex This class is responsible for downloading the drug indexes

DownloadService RobospiceService class for handling the download of data. This is an uncached service as the data is downloaded infrequently, so the cache will most likely have expired by the next time the request is made.

NHS Wales Injectable Medicines Guide

Report Name	Test Results
Author (User Id)	Aidan Wynne Fewster (awf1)
Supervisor (User Id)	Andrew Starr (aos)
Module	CS39440
Degree Scheme	G400 (Computer Science)
Date	20th April 2014
Revision	1.0
Status	Release

This document will provide the result for any testing that was carried out on the application. It will include the results of the unit tests, the results of the intergration tests and the results of stress testing the application.

1 Intergration Tests

To test that the application works as expected a list of integration tests were made and then ran on two devices, one device running API version 19 and the other running API version 8 Using two devices, with varying ages increases the accuracy of the results. To further increase the accuracy of the results the tests would be executed on more devices.

Test	Expected result	API 8 result	API 19 result
Login Activity			
Enters invalid password and attempt to login	User is notified via toast that username is incorrect	**PASS**	**PASS**
Attempt to login whilst in flight mode	User is notified of connection error	**PASS**	**PASS**
Attempt to login using correct details	Download activity is launched	**PASS**	**PASS**
		PASS	**PASS**
Download activity		**PASS**	**PASS**
User presses back button	Return to login screen	**PASS**	**PASS**
User minimises application and re-enters	Download continues in background	**PASS**	**PASS**
Calculator download fails	Dialog asking the user whether they would like to retry is displayed	**PASS**	**PASS**
Index download fails	Dialog asking the user whether they would like to retry is displayed	**PASS**	**PASS**
Drug information download fails	Dialog asking the user whether they would like to retry is displayed	**PASS**	**PASS**
User clicks the retry button	The appropriate download task is restarted	**PASS**	**PASS**
User clicks cancel button	User is logged out and login activity is displayed	**PASS**	**PASS**
Main activity			
User presses back button	Close application	**PASS**	**PASS**
Check last update date is set correctly	The date within the last updated TextView contains the date the database was last updated	**PASS**	**PASS**
Browse drugs button press	Open the browser drugs activity	**PASS**	**PASS**
Browse calculator button presses	Open the browser calculator activity	**PASS**	**PASS**
Update button pressed	Open the download activity	**PASS**	**PASS**

Browse drugs			
Drugs are displayed properly	Full list of available drugs are displayed within the list	**PASS**	**PASS**
User enters text into the search box	The lists of drugs are filtered by the inputted text	**PASS**	**PASS**
User clicks drug	View drug activity is launched for that drug	**PASS**	**PASS**
Browse calculators			
Drugs are displayed properly	Full list of drugs with calculators are displayed within the list	**PASS**	**PASS**
User enters text into the search box	The lists of drugs are filtered by the inputted text	**PASS**	**PASS**
User clicks drug	Calculator is launched for that drug	**PASS**	**PASS**
View drug activity			
The correct drug is displayed	The drug selected by the user is displayed	**PASS**	**PASS**
User clicks on header helper button	The helping information for the header is displayed.	**PASS**	**PASS**
User clicks calculator button	The calculator activity is opened	**PASS**	**PASS**
Calculator test			
A dosage calculation using correct values is performed by the user	The results of the calculation and an explanation for the equation used is displayed to the user	**PASS**	**PASS**
An infusion rate calculation using correct values is performed by the user	The results of the calculation and an explanation for the equation used is displayed to the user	**PASS**	**PASS**
User enter 0kg for weight	Error about weight is displayed to user	**PASS**	**PASS**
User leaves weight field empty	Error about weight is displayed to user	**PASS**	**PASS**
User enters weight of 5kg	A warning is displayed to the user regarding the weight	**PASS**	**PASS**
User enters weight of 500kg	A warning is displayed to the user regarding the weight	**PASS**	**PASS**
User enters 0 for dosage	Error about dosage is displayed to user	**PASS**	**PASS**
User leaves dosage field empty	Error about dosage is displayed to user	**PASS**	**PASS**
User enters 0 for time	Error about time is displayed to user	**PASS**	**PASS**

User leaves time field empty	Error about time is displayed to user	**PASS**	**PASS**
User enters 0 for concentration	Error about concentration is displayed to user	**PASS**	**PASS**
User leaves concentration field empty	Error about concentration is displayed to user	**PASS**	**PASS**
Common			
Exit menu item pressed	Application is terminated	**PASS**	**PASS**
Logout menu item pressed	User is logged out and then the login activity is opened	**PASS**	**PASS**
Home menu item pressed	The main activity is launched	**PASS**	**PASS**
Update item pressed	The download activity is launched	**PASS**	**PASS**
Update item pressed	The download activity is launched	**PASS**	**PASS**
Browse drugs item pressed	The browse drugs activity is launched	**PASS**	**PASS**
Browse calculator pressed	The browser calcualtor activity is launched	**PASS**	**PASS**

2 JUnit test results

Unit tests were written for every class of the application, testing each public and protected method. Most tests contained multiple assertions, testing that the expected output was returned when correct information is entered and that an error is raised when the incorrect data is entered.

The following pages contain the results of the Unit tests.

Tests: 107 total, 107 passed

16.40 s

com.fewstera.injectablemedicinesguide.database.tests.DatabaseHelperTest

1.19 s

testCreateDrug	passed	126 ms
testCreateDrugCalcInfo	passed	127 ms
testCreateDrugIndex	passed	126 ms
testGetAllDrugIndexes	passed	76 ms
testGetAllDrugsWithCalcs	passed	101 ms
testGetDrugCalcInfoFromDrugId	passed	102 ms
testGetDrugFromId	passed	102 ms
testGetDrugInformationsFromDrugId	passed	100 ms
testTruncateAll	passed	126 ms
testTruncateCalcs	passed	102 ms
testTruncateIndexs	passed	100 ms

com.fewstera.injectablemedicinesguide.models.DrugCalculatorInfoTest

0 ms

testGetConcentrationUnits	passed	0 ms
testGetDoseUnits	passed	0 ms
testGetDrugId	passed	0 ms
testGetInfusionRateLabel	passed	0 ms
testGetInfusionRateUnits	passed	0 ms

com.fewstera.injectablemedicinesguide.models.tests.DrugCalculatorInfoTest

25 ms

testConstuct	passed	0 ms
testGetAndSetConcentrationUnits	passed	0 ms
testGetAndSetDoseUnits	passed	0 ms
testGetAndSetDrugId	passed	0 ms
testGetAndSetFactor	passed	0 ms
testGetAndSetInfusionRateLabel	passed	0 ms
testGetAndSetInfusionRateUnits	passed	0 ms
testGetAndSetTimeRequired	passed	0 ms
testGetAndSetWeightRequired	passed	25 ms

com.fewstera.injectablemedicinesguide.models.tests.DrugIndexTest		0 ms
testConstuct	passed	0 ms
testGetId	passed	0 ms
testGetName	passed	0 ms

com.fewstera.injectablemedicinesguide.models.tests.DrugInformationTest		0 ms
testConstuct	passed	0 ms
testGetHeaderHelp	passed	0 ms
testGetHeaderText	passed	0 ms
testGetSectionText	passed	0 ms

com.fewstera.injectablemedicinesguide.models.tests.DrugTest		0 ms
testAddAndGetDrugInformation	passed	0 ms
testConstuct	passed	0 ms
testEmptyDrugInfos	passed	0 ms
testGetAndSetId	passed	0 ms
testGetAndSetName	passed	0 ms
testToString	passed	0 ms

com.fewstera.injectablemedicinesguide.tests.AuthTest		254 ms
testConstuct	passed	0 ms
testIsLogged	passed	25 ms
testIsValid	passed	204 ms
testLogout	passed	0 ms
testPrepareUrl	passed	0 ms
testSaveCredentialsAndGetters	passed	25 ms
testSetCredentials	passed	0 ms

com.fewstera.injectablemedicinesguide.tests.BrowseDrugsActivityTest		1.70 s
testListOnScreen	passed	483 ms
testListSize	passed	407 ms
testPreconditions	passed	482 ms
testSearchOnScreen	passed	328 ms

com.fewstera.injectablemedicinesguide.tests.CalcDrugSelectActivityTest		1.86 s

testListOnScreen	passed	532 ms
testListSize	passed	433 ms
testPreconditions	passed	358 ms
testSearchOnScreen	passed	532 ms

com.fewstera.injectablemedicinesguide.tests.CalculateActivityTest 3.97 s

testButton	passed	510 ms
testConcentration	passed	510 ms
testDose	passed	407 ms
testDrugHeader	passed	433 ms
testInfusionRate	passed	431 ms
testPreconditions	passed	357 ms
testSpinner	passed	483 ms
testTime	passed	430 ms
testWeight	passed	406 ms

com.fewstera.injectablemedicinesguide.tests.CalculatorTest 1 ms

testAdrenalineCalcualtions	passed	0 ms
testGlycerylTrinitrate	passed	0 ms
testMidazolamCalcualtions	passed	0 ms
testSetAndGetConcentration	passed	1 ms
testSetAndGetDose	passed	0 ms
testSetAndGetInfusionRate	passed	0 ms
testSetAndGetTime	passed	0 ms
testSetAndGetType	passed	0 ms
testSetAndGetWeight	passed	0 ms
testValidateConcentration	passed	0 ms
testValidateErrorDose	passed	0 ms
testValidateErrorInfusionRate	passed	0 ms
testValidateErrorWeight	passed	0 ms
testValidateSuccess	passed	0 ms
testValidateTime	passed	0 ms
testValidateWarnWeight	passed	0 ms

com.fewstera.injectablemedicinesguide.tests.DownloadDataActivityTest 1.45 s

testHeader	passed	306 ms

testMessage	passed	281 ms
testPreconditions	passed	280 ms
testProgressBar	passed	280 ms
testProgressMessage	passed	305 ms

com.fewstera.injectablemedicinesguide.tests.DrugTest 0 ms

testAddDrugInformation	passed	0 ms
testGetDrugInformations	passed	0 ms
testGetId	passed	0 ms
testGetName	passed	0 ms
testToString	passed	0 ms

com.fewstera.injectablemedicinesguide.tests.LoginActivityTest 1.09 s

testLoginButton	passed	279 ms
testPasswordTextView	passed	306 ms
testPreconditions	passed	305 ms
testUsernameTextView	passed	204 ms

com.fewstera.injectablemedicinesguide.tests.MainActivityTest 2.26 s

testBrowseButton	passed	429 ms
testCalculatorButton	passed	331 ms
testPreconditions	passed	460 ms
testUpdateButton	passed	381 ms
testUpdateText	passed	306 ms
testWelcomeText	passed	358 ms

com.fewstera.injectablemedicinesguide.tests.PreferencesTest 52 ms

testDelete	passed	26 ms
testDownloadCompleteBool	passed	0 ms
testSetAndGetString	passed	26 ms

com.fewstera.injectablemedicinesguide.tests.ViewDrugActivityTest 2.54 s

testDrugInfoContent	passed	432 ms
testDrugInfoHeaders	passed	407 ms
testDrugInfoHelper	passed	432 ms
testHeaderTextView	passed	434 ms
testPreconditions	passed	460 ms
testTitle	passed	380 ms

3 Exerciser monkey stress text

Exerciser Monkey is a tool provided with the Android SDK, which is used for stress testing Android applications. The tool simulates a set amount of random events on the device, such as button presses, screen presses, volume changes and screen rotations. The tests are used to ensure that applications run well under stressful tasks and that parts of the application do not throw errors.

It was planned that the application would be installed on a device, then using Exerciser Monkey, execute 5000 random events to the device.

Below is the result of executing 5000 random events to the device.

```
$ adb shell monkey -p com.fewstera.injectablemedicinesguide 5000
    // activityResuming(com.fewstera.injectablemedicinesguide)
    // activityResuming(com.fewstera.injectablemedicinesguide)
    // activityResuming(com.fewstera.injectablemedicinesguide)
    // activityResuming(com.fewstera.injectablemedicinesguide)
    // activityResuming(com.fewstera.injectablemedicinesguide)
    // activityResuming(com.android.launcher)
    // activityResuming(com.android.launcher)
    // activityResuming(com.android.launcher)
    // activityResuming(com.android.launcher)
    // activityResuming(com.android.launcher)
Events injected: 5000
## Network stats: elapsed time=11069ms (0ms mobile, 11065ms wifi, 4ms not connec
```

The tests results show that the test was successful as no time-out warnings, exceptions or errors were thrown by the debugging console.

End of appendices

Annotated Bibliography

[1] "What are the benefits of mvc?" http://blog.iandavis.com/2008/12/09/what-are-the-benefits-of-mvc/, 2008.

> Model-View-Controller is a popular design pattern that I am familar with. This document discusses the benefits of MVC.

[2] "Extreme Programming: A gentle introduction," http://www.extremeprogramming.org/, 2010.

> Extreme programming (XP) was the methodology that I first planned to use for this project. It uses 12 practices each of which would've benefited the project.

[3] "Waterfall model," http://c2.com/cgi/wiki?WaterFall, 2011.

> The waterfall model is a popular methodology used by a large amount of software projects. The waterfall model was the methodology that was used throughout this project

[4] "Singleton design pattern in java," http://howtodoinjava.com/2012/10/22/singleton-design-pattern-in-java/, 2012.

> This blog post discusses singletons within the Java programming language

[5] "PhoneGap API Documentation - Local storage," http://stackoverflow.com/questions/14210832/can-phonegap-app-work-in-background, January 2013.

> StackOverflow question asking how to implement background process within phonegap

[6] "Activity testing," http://developer.android.com/tools/testing/activity_testing.html, 2014.

> A guide on creating tests for Android Activities

[7] "Activity testing," http://developer.android.com/tools/help/monkey.html, 2014.

> Android documentation on using the Application Exerciser Monkey tool, to stress test applications

[8] "Alertdialog- android developers," http://developer.android.com/reference/android/app/AlertDialog.html, 2014.

> Alert Dialogs were used throughout the project to notify the user of an action. This is the Android documentation on AlertDialogs.

[9] “Android,” http://www.android.com/, 2014.

Android is the operating system that has been used to create the application.

[10] “Android and iOS Continue to Dominate the Worldwide Smartphone Market with Android Shipments Just Shy of 800 Million in 2013,” http://www.apple.com/uk/ios/, February 2014.

This press release states the 2013 market share for a variety of devices and operating systems. This was used when determining the most suitable platform.

[11] “Android code style guidelines,” https://source.android.com/source/code-style.html, 2014.

Android specification on their coding style gudielines.

[12] “Android developer design guidelines,” https://developer.android.com/design/index.html, 2014.

Android design guidelines were used throughout the design and implementation stages, to ensure the user interface followed Android guidelines.

[13] “Android SDK,” http://developer.android.com/sdk/, 2014.

A brief introduction to the Android SDK, providing links to other useful resources.

[14] “Android studio,” http://developer.android.com/sdk/installing/studio.html, 2014.

Android Studio is a new IDE created by Android specifically for creating Android applications.

[15] “Apache 2.0 licence,” http://www.apache.org/licenses/LICENSE-2.0.html, 2014.

The Apache 2.0 licence is an open source software licence, that allows the users to modify and resell the product, as long as the original notice is kept.

[16] “Apple (United Kingdom) - iOS 7,” http://www.apple.com/uk/ios/, 2014.

Apple’s iOS is the operating system ran on the popular iPhone device, as well as many other devices such as the iPad and iPod.

[17] “Asynctask - android documentation,” http://developer.android.com/reference/android/os/AsyncTask.html, 2014.

The Android SDK provides a class for executing tasks outside of the main thread called AsyncTask. This is the documentation for AsyncTasks.

[18] “Eclipse ide,” https://www.eclipse.org/, 2014.

Eclipse is an open source IDE. There is a plugin made for Eclipse allowing it to build Android applications

[19] “Electromagnetic interference : MHRA,” http://www.mhra.gov.uk/Safetyinformation/Generalsafetyinformationandadvice/Technicalinformation/Electromagneticinterference/, 2014.

A document outling the possiblities of mobile interference on medical equipment. This is the reason why mobile offline access most be implemented into the final application.

[20] “Electromagnetic interference : MHRA,” http://www.mhra.gov.uk/Safetyinformation/Generalsafetyinformationandadvice/Technicalinformation/Electromagneticinterference/, 2014.

Medua is the desktop service that the NHS currently use to acces the database. Medua what was expected of the final application. The Medua website was also used to test the calculator page.

[21] “Froyo realse notes,” http://android-developers.blogspot.co.uk/2010/05/android-22-and-developers-goodies.html, 2014.

Froyo, the name of the Android SDK API version 8, this is the minimum SDK version supported by this application

[22] “Genymotion,” http://www.genymotion.com/, 2014.

Genymotion is an Android emulator. Genymotion is quicker than the standard Android emulator and allows you to easily create emulators of the most populat devices.

[23] “Git,” http://git-scm.com/, 2014.

Git is a distributed version control system. Git was used throughout the project to keep a log of all changes. If there was ever a problem with the project, the project could be rolled back to an earlier version

[24] “Github,” http://github.com/, 2014.

Github is service used for storing git repositories. Github provide a free account to students.

[25] “GLYPHICONS - library of precisely prepared monochromatic icons and symbols.” http://glyphicons.com/, 2014.

Glyphicon is a set of icons released under the Apache 2 licence. This library wasy used for some iconogprahy within the application

[26] “Gradle,” http://www.gradle.org/overview, 2014.

Grade is an automatic build configuration tool, similar to Apaches Maven. Gradle allows you create custom build configurations for a variety of setups.

[27] “Icon search engine,” http://findicons.com/, 2014.

A search engine built for finding icons. This was used to find a open source icon for the applications icon.

[28] “Introduction to Test Driven Development (TDD),” http://www.agiledata.org/essays/tdd.html, 2014.

Test driven development is the procress of creating software, were the tests for the software are written before the code is implemented. Test driven development was used in parts of this project.

[29] "Java Programming Language," http://www.java.com/, 2014.

The Java programming language is the base language used throughout Android development.

[30] "Javadoc documentation," http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html, 2014.

Javadoc, a tool and style used for commenting Java code

[31] "JUnit Testing," http://junit.org/, 2014.

JUnit is a Java framework for testing individual units. JUnit tests have been built for each class within the application

[32] "Lucid charts," http://lucidcharts.com/, 2014.

Lucid charts is a HTML 5 UML creator. I used this to produce the mock-ups and class diagrams

[33] "Monograph format," http://www.empr.com/drug-monograph-format/section/793/, 2014.

This page shows an example of what a drug monograph should look like, this was used to allow me to research what would be created, before taking on the project

[34] "NHS Choices - Your health, your choices," http://www.nhs.uk/Pages/HomePage.aspx, 2014.

This is the homepage of the National Health Service, whom this project was created for.

[35] "Phonegap — Home," http://phonegap.com/, 2014.

Phonegap is a multi-platform hybrid application builder. Phonegap allows you to create native applications for multiple platforms using HTML, CSS and Javascript.

[36] "PhoneGap API Documentation - Local storage," http://docs.phonegap.com/en/3.0.0/cordova_storage_storage.md.html, 2014.

Documentation for Phonegaps local storage functionality. Local storage allows the hybrid application to store information on the devices local disc.

[37] "Progaurd," http://developer.android.com/tools/help/proguard.html, 2014.

ProGaurd, a feature provided by Android SDK, allowing developers to protect and optimise their applications code.

[38] "Programming with Objective-C," https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html, February 2014.

A brief overview of the objective-C language and how it relates to other languages like C and C++.

[39] “Robospice,” https://github.com/stephanenicolas/robospice, 2014.

Robospice is a library created for Android development, that allows developers to easily create asynchronous running services.

[40] “Robotium black-box testing,” https://code.google.com/p/robotium/, 2014.

Robotium is an Android library which allows developers to white box test applications easily.

[41] “Service - android documentation,” http://developer.android.com/reference/android/app/Service.html, 2014.

The Android SDK provides a class for executing long running tasks outside of the main thread called a Service. This is the documentation for an Android Service.

[42] “Sharedpreferences — android developers,” http://developer.android.com/reference/android/content/SharedPreferences.html, 2014.

A shared preference is a key-value pair accessed by providing the key as a string. SharedPreferences are useful for storing single pieces of information on the device. This the documentation for shared preferences

[43] “Software Alliance Wales,” http://softwarealliancewales.com/, 2014.

The Software Alliance is who provided this project to the Abersywtyth University.

[44] “SQLite,” http://www.sqlite.org/, 2014.

An SQLite database was used within the project to store the data downloaded from the server.

[45] “SQLite Datatypes,” http://www.sqlite.org/datatype3.html, 2014.

This is the list of available datatypes within SQLite databases

[46] “Stack Overflow,” http://stackoverflow.com/, 2014.

StackOverflow was used throughout the project to check on best practices and to solve common problems.

[47] “Strings.xml resouce,” http://developer.android.com/guide/topics/resources/string-resource.html, 2014.

The Android SDK provides a XML file that is used to store all strings used for the project. This is the Android documentation on strings.xml

[48] “SVN - Subversion,” http://subversion.tigris.org/, 2014.

SVN is a version control system that was codcidered during the background stage of the project.

[49] "W3 XML Specification," http://www.w3.org/XML/, 2014.

This is the W3 specification for XML. This was used as the data is provded in XML

[50] "Web applications (WEBAPPS) ," http://www.w3.org/2008/webapps/, 2014.

This is the W3 specification group on web applications.

[51] AppBuilder, "What is a Hybrid Mobile App?" http://phonegap.com/, June 2012.

Blog post explaining what a hybrid mobile application is.