

|  |
| --- |
| School of Computing  Faculty of Engineering |

Full Title of Project

Full Name of Author

Submitted in accordance with the requirements for the degree of  
<Name of Degree>

Session (e.g. 2014/2015)

The candidate confirms that the following have been submitted*:*

*<As an example>*

|  |  |  |
| --- | --- | --- |
| **Items** | **Format** | **Recipient(s) and Date** |
| *Deliverables 1, 2, 3* | *Report* | *SSO (xx/xx/xx)* |
| *Participant consent forms* | *Signed forms in envelop* | *SSO (xx/xx/xx)* |
| *Deliverable 4* | *Software codes or URL* | *Supervisor, assessor (xx/xx/xx)* |
| *Deliverable 5* | *User manuals* | *Client, supervisor (xx/xx/xx)* |

Type of Project: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)

© <Year of Submission> The University of Leeds and <full name of candidate>

# Summary

*<Concise statement of the problem you intended to solve and main achievements (no more than one A4 page)>*

# Acknowledgements

*< Karim Djememe This page should contain any acknowledgements to those who have assisted with your work. Where you have worked as part of a team, you should, where appropriate, reference to any contribution made by others to the project.*

*Note that it is not acceptable to solicit assistance on ‘proof reading’ which is defined as “the systematic checking and identification of errors in spelling, punctuation, grammar and sentence construction, formatting and layout in the text”; see* [*http://www.leeds.ac.uk/qat/documents/policy/Proof-reading-policy.pdf*](http://www.leeds.ac.uk/qat/documents/policy/Proof-reading-policy.pdf)*. >*

# Table of Contents

Summary iii

Acknowledgements iv

Table of Contents v

Introduction 1 1

1.1 Project Overview 1

1.1.1 Problem statement. 1

1.2 Minimum Requirements 1

1.3 Objectives 1

1.4 User Collaborative Agile Design Methodology 1

1.5 Schedule 1

2 Background Research 2

2.1 Mobile Operating Systems 2

2.2 Programming Language For Development 2

Objective C 2

2.3 REST Frameworks 2

2.4 XML Conversion Libraries 2

2.5 Integrated Development Environment 3

2.5.1 IOS 3

2.5.2 Java 3

2.6 Database Design 3

3 Design 4

3.1 Gathering Requirements 4

3.2 Low Fidelity Prototype 4

3.3 Clients Reflection on Prototype 4

3.4 User Interface Design 4

3.5 Mobile Application Structure 4

3.6 REST Server Structure & Administration User Interface 4

3.7 Database Structure 4

Chapter 2 Tables and Figures 6

2.1 Tables using the ‘table caption’ and ‘table description’ Styles 6

2.2 Figures using the ‘figure caption’ and ‘figure description’ Styles 6

4 Implementation 7

4.1 Iteration 1 7

4.2 Iteration 2 7

4.3 Iteration 3 8

4.4 Iteration 4 8

List of References 9

Appendix A External Materials 10

A.1 Level 2 Heading with ‘heading 2’ Style Applied by Pressing Ctrl Shift 2 10

A.1.1 Level 3 Heading with ‘heading 3’ Style Applied by Pressing Ctrl Shift 3 10

A.1.1.1 Level 4 Heading with ‘heading 4’ Style Applied by Pressing Ctrl Shift 4 10

Appendix B Ethical Issues Addressed 11

B.1 Level 2 Heading 11

# Introduction 1

Text under chapter heading. Text under chapter heading. Text under chapter heading. Text under chapter heading. Text under chapter heading. Text under chapter heading. Text under chapter heading.[[1]](#footnote-1)

## 1.1 Project Overview

Text under level 2 heading. Text under level 2 heading. Text under level 2 heading. Text under level 2 heading.

### 1.1.1 Problem statement.

Text under level 3 heading. Text under level 3 heading. Text under level 3 heading. Text under level 3 heading.

## 1.2 Minimum Requirements

## 1.3 Objectives

## 1.4 User Collaborative Agile Design Methodology

For the development process of the project and Agile methodology was followed. As the project has to be completed in quite a short duration of time the ability to provide working software at each stage of development. Constant interaction with the client is allows for changes during implementation at any time so new directions can be taken with little effort. This is advantageous as following this methodology gives the ability deliver something that the client can use even if there are unforeseen problems or large changes that affect the schedule of the project.

The main goals of this methodology are :

* Get feed back from client by continuously delivering working software in each of the iterations.
* Allow changes at any time during development.
* Work directly with the client face-to-face if possible to deliver the best possible product to the customer.
* Justifying design/development choices and following best practices to speed up development process. (The Agile Alliance , 2015)

### 1.4.1 Agile Story Workshops

Various techniques are used in Agile, such as Agile story workshops. A story workshop is meeting with developers and the client(s) where stories are created to explain what is needed for each feature. For example “As a user, I want to be able to see information about a drug at a glance. This includes; a name, brand names, dose, type of drug, interactions, side effects and various routes of administration for paediatric and adult patients.” This describes the feature in what information needs to be displayed for a drug. This is done for all users so example of a different user would be “As an admin user, I want to be able to add or remove the drugs that are displayed on the mobile app.”. This describes the feature of adding an administration interface to be able to be able to add and remove drugs. This allows the developer(s) and client(s) to prioritise the features easily by rating the importance of each story. The developer(s) then plan which features to be done in each iteration make sure at the end of each iteration that the software could be shipped if necessary. (Quotient Integrated Solutions, 2014).

For this project this technique was followed but with the use of storyboards as well. From the user stories, storyboards were created to show a visual process of how the user interface. This allowed the client to see how the features would look and if this is how it was imagined in the user stories. This speeds up development times as the storyboards can help confirm the designs of the basic user interface eliminating the risk of the user interface needing large changes that could affect the schedule of the project. This can be done quickly and simply on paper in preparation or during a meeting. Following this approach conforms to principle 10 of The Agile Alliance’s Principles of agile software “Simplicity--the art of maximizing the amount of work not done--is essential” (The Agile Alliance , 2015). As the work that would be created in constantly changing the user interface to suit the client’s imagination would be eliminated because it was already confirmed.

## 1.5 Schedule

# 2 Background Research

## 2.1 Developing For Android Or IOS

The two most popular mobile operating systems are Android and IOS. There is debate on which is truly the most popular but mostly all sources agree these are the two market leaders. The two top companies for gathering data on the popularity of mobile operating systems are StatCounter and Net Market Share. Net Market Share gathers its statistics by measures total traffic and StatCounter measures daily unique users (Bott, 2014). Statistics for this year from; (Net Market Share, 2015), (StatsCounter, 2015), are shown in figure 2.1.1.

### Figure 2.1.1

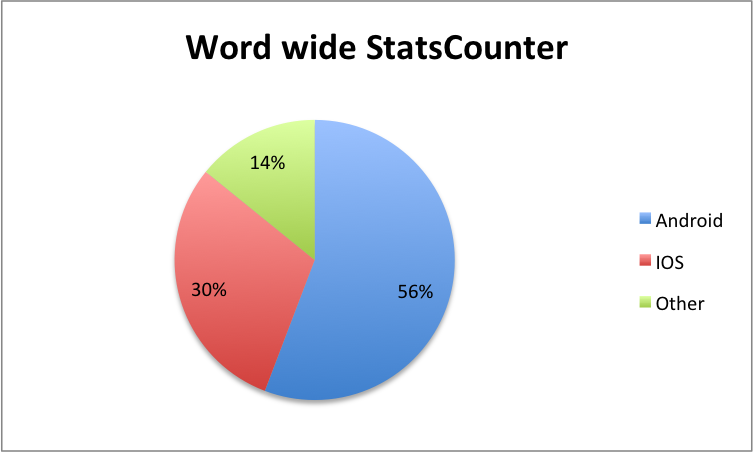
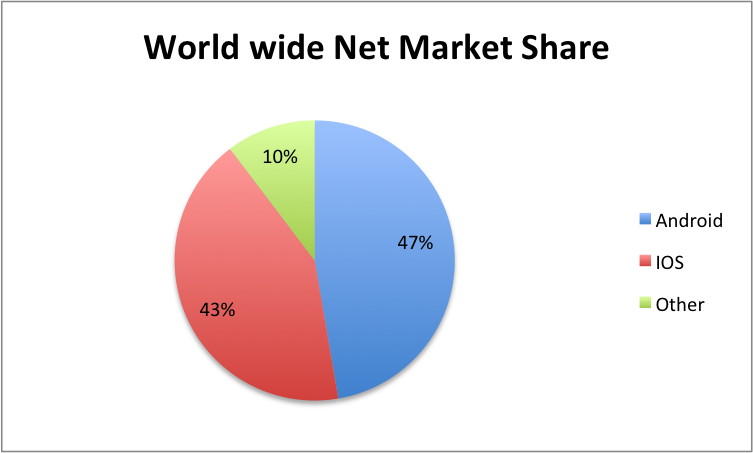
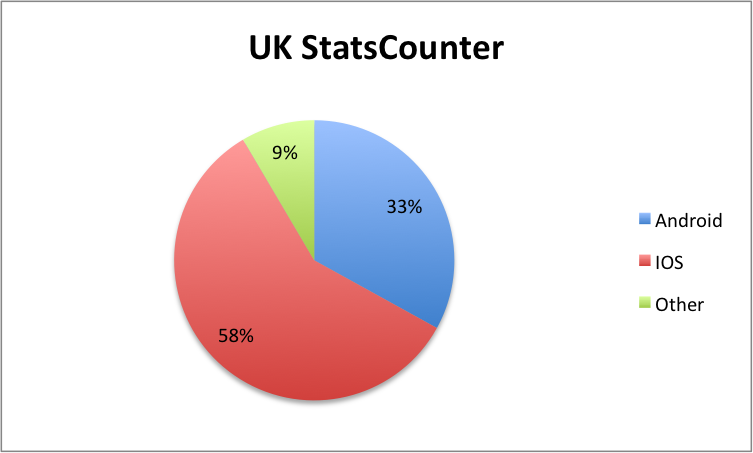


Figure 2.1.1 shows the percentages of market share for the mobile operating systems Android and IOS. As seen Android is the dominant operating system from both sources.

### Figure 2.1.2



Although the two sources agree that Android is the dominant operating system world wide as seen in figure 2.1.1, when filtered for the UK as shown in figure 2.1.2 IOS the dominant mobile OS in the UK (StatsCounter, 2015). Net Market Share does not offer region filtering for free so it is assumed that the statistics follow the same pattern.

### 2.1.1 Android

Android is claimed to be the most popular mobile operating system world wide and provides a very open market place to distribute apps (Android, 2015). Android has many devices in which its operating system runs on from various companies such as Samsung, HTC, Motorola, ASUS and many others. The devices created by these companies operate on many different screen sizes and versions of the operating systems. A download is available from Google Play’s support page that gives an A-Z list of all the devices that Android supports, the number of devices is approximately 8,725 (Google, 2015). As there are a great deal of devices testing can be a long process when developing for Android.

Google Play is Android’s main source of mobile application distribution although Android does not restrict developers to publish Android applications on other application stores. To publish an application a registration fee of $25 is required this is a one time fee per account and allows developers to publish their applications on the Google Play store. Google allows Android applications to be uploaded by developers on to the Play Store in short time usually the same day as it was published (Google, 2015). The applications are scanned automatically for malicious and poorly implemented software as they are uploaded to maintain quality applications on the application store (Mills, 2012). This is a major benefit to developers as the software can be released very quickly minimizing the gap between development and release lowering the cost of development especially for companies that hire many developers for a particular application.

### 2.1.2 IOS

The usage of IOS devices is prominent in the UK with over 50% of data traffic on IOS (StatsCounter, 2015). In comparison to Android there are far fewer device variants on IOS, various versions of iPad, iPhone, iPod Touch in which under 20 devices run IOS 8.0. As there is such a small number of devices testing is far quicker as well as the there is an emulator for each device build into the IDE that Apple provides.

The Apple App store is the only place where applications can be downloaded to mobile devices. The development for IOS is much more restricted than Android. To publish an application to the App store an annual fee of $99 is required to enrol on the developer program. This then gives the developer a certificate to authorise the apps they develop. Once the application is developed it has to go through the review process. This takes an average of 8 days according to the crowd sourced site (Shiny Development, 2015) as of 14th of April. If the application is not rejected it then will be released on to the application store, although if it is rejected the entire review process has to be repeated once the issue is resolved (Apple, 2015). In comparison to Android the release time can take much longer as if an app especially if some guidelines are missed during development. The reason for Apple doing this is to ensure the quality of the applications on its App store to keep the companies good reputation. This can be frustrating to developers and seriously impact development times but ensures only higher quality applications can be released.

### Conclusion to OS choice for development

The choice of operating system to develop for is IOS for the following reasons:

* The client wanted the development to be in IOS. It is the clients opinion that is IOS the most popular among the student doctors in Leeds University.
* IOS is assumed to be the most popular in the UK for mobile devices as discussed above.
* Testing an Android application may take longer than IOS due to the amount of variation between the many devices that use Android. Due to the limited time of the project, if Android was chosen the application may not be compatible with some devices as they cannot all be tested within the time frame. IOS however can has very few devices and the XCode IDE has emulator for all the most recent devices.
* Although deploying the application on the App store may take longer than Android, the App Store review process will help identify any unforeseen bugs in the application and when it gets through it’s a sign that this application is of high quality.

## 2.2 Programming Language For Development

### Objective C

The chosen language for the mobile application is Objective C this is because it is the native language for Apple’s operating systems and is has a large supported community. This means it has substantial examples and libraries to help the development of this project.

Swift was an alternative but as it is still very new there could be issues with constant updates and changes causing features to become outdated or cause bugs (Eid, 2014). Also because it is so new there may not be as many examples to learn the code as compared with Objective C. Although this language is simpler syntactically compared to Objective C, Swift still uses Objective C libraries. This could make the code difficult to understand if written in two different languages.

C++/C could have also been used but the code would have needed some Objective C wrappers and converters for the application to run on IOS. This would of meant mixing languages making the code less readable and verbose.

## 2.3 REST

Representational state transfer, also known as “REST” is an architectural style of developing a distributed system which was first shown in a paper by Roy Fielding. As REST is style not a standard, there is no W3C recommendation for it and it is simple enough to be used for a variety of approaches. (Vogel, 2014) (Elkstein)

### 2.3.1 Why Use REST?

Having a RESTful layer means that any application from any operating system can communicate with the server meaning that if needed other developers can look at the API and create their own app that displays the data the way they want it. REST is stateless meaning doesn’t rely on traditional methods of communicating with a server as it doesn’t create a session of persistent connections, it just uses HTTP perform operations on a web service. The benefit of this is speed and scalability as the application can just use a URI to get the exact data without having to maintain a session. Another advantage is loose coupling which means it can be platform independent as HTTP can communicate with mostly all devices that can connect to the internet. The client wants this system to be used after the project has been completed. Developing the server using REST will allow the data to be transferred to the app as it does not need to know anything about the server other than the URI for it to understand the data. Any application with an internet connection and XML handling can get and understand this data. If the app needs changing due to updates released by the operating system it can be completely redesigned without having to do anything to the server. This data will be in XML format so any platform can use the data to display the information.

### 2.3.2 REST Framework Jersey

To make the development of the RESTful server easier a framework called Jersey was used. There are many different frameworks that can be used to develop REST services but ///I HAVE HAD EXPERIENCE WITH JERSEY AND THE IDE ECLIPSE DURING MY DEGREE////. Due to the short time of the project the decision to use a more familiar framework will help shorten development and research time. Jersey is also recommended by IMB to create REST services with the use of Apache Tomcat for the container of the application (IMB, 2009).

Jersey also has some useful features built in such JAXB which converts nested Java classes to be converted into XML using simple notation. This a very useful library as the XML generation and input can be done automatically with a small amount of code. As this method has been tried and tested it reduces the risk of bugs in comparison to implementing the marshalling from scratch.

Jersey is designed to work with the Apache Web container Tomcat. The need for a web container is to run the web application and to handle the dynamic webpages as the server itself can only return static pages. Apache Tomcat will be used to precompile/validate the web application and offers a GUI for to do this simply. This will speed up the process for testing different versions of the server to make sure everything is working properly.

## 2.5 Integrated Development Environment

### 2.5.1 XCode

The chosen IDE to develop the mobile application is XCode as it provides all the necessary tools and features needed to create mobile applications in IOS. XCode is what Apple recommends to develop IOS applications and Apple includes a great deal of documentation of how to develop applications for IOS in XCode. This IDE can understand languages of C/C++/Obective C and Swift and includes emulators for the most popular mobile devices to test the applications. XCode also includes build in Git Hub version control which can control the changes to the project (Apple Developer 2015). XCode also has a direct link to the certificate control for the Apple App Store so this will enable the project to be deployed easily when run with a developers account. This will make it easier for the client to get it on to the application store at the end of the project.

### 2.5.2 Eclipse

The chosen IDE to develop the server is Eclipse. Eclipse has the ability add different plugins internally in the IDE so it is very easy to customise for development of a particular application. The Eclipse IDE is also used in many tutorials on how to develop web applications for REST and what was used in the Distributed Systems module Lab sessions of the IT degree. This choice will help shorten the time having to familiarise with interfaces/features and increase time on actual development.

## 2.6 Database Design

//DOOOO LATER MAYBE MOVE TO DESIGN SECTION

One of the requirements for the project is to have a database to store all the data on the drugs and pathogens. The client already has a database with drug data but nothing on the pathogens.

Mysql,WAMP, JDBC

# 3 Design

## 3.1 Gathering Requirements

Client meetings, object structure, process of getting minimum requirements.

## 3.2 Low Fidelity Prototype

Paper to white board meetings

## 3.3 Clients Reflection on Prototype

Reviews on intial design changes

## 3.4 User Interface Design

Show storyboard

## 3.5 Mobile Application Structure

Basic IOS didFinishWithOptions() onLoad() etc.

## 3.6 REST Server Structure & Administration User Interface

Methods to implement server return html,xml, delete, add, EDIT?

## 3.7 Database Structure

## 

# 4 Implementation

* The following segment of the report describes how the mobile application, server and database were implemented. The implementation section will give information on process of development to solve the project problem. The agile methodology is used for this project thus the implementation section is split into different iterations. At the end of each iteration a demonstration of the working software is given with an evaluation of work done during that iteration. This was done to simulate the part of the agile methodology “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale” (The Agile Alliance , 2015).

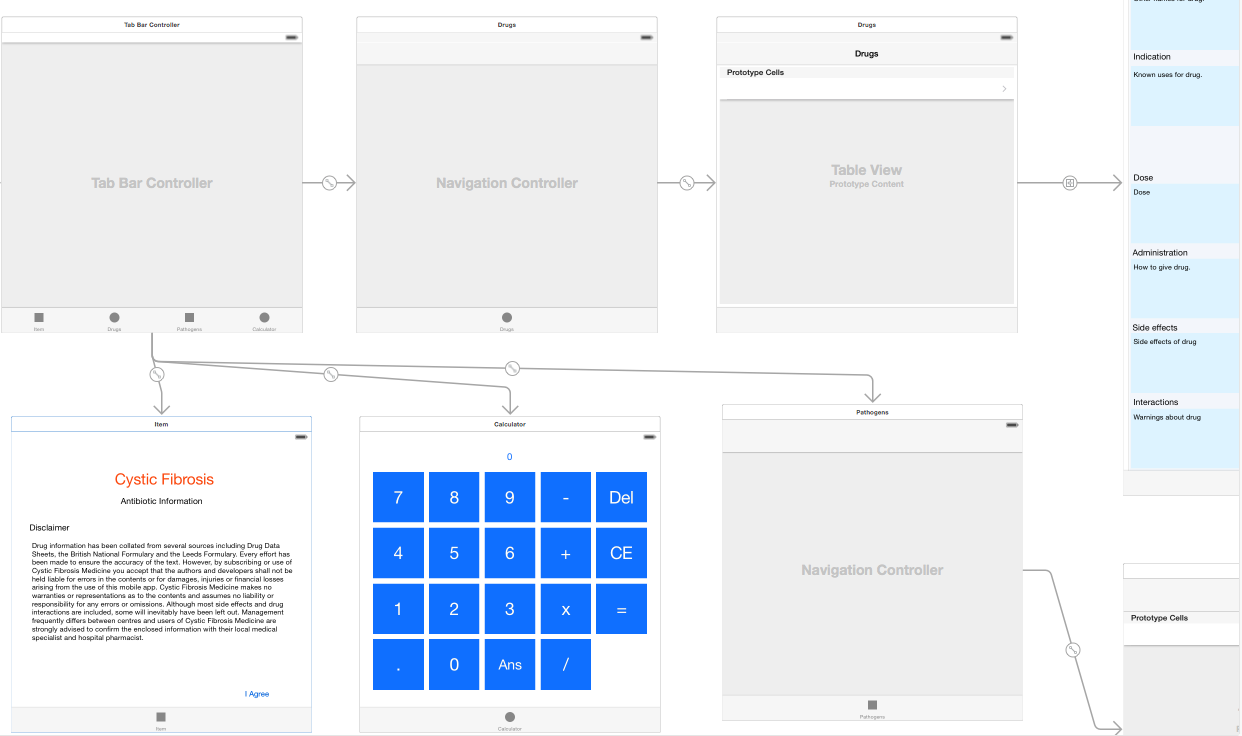
## 4.1 Iteration 1

### 4.1.1 User Interface

The aim of this iteration was implement a working prototype of the mobile application. The first stage involved creating the user interface in the Interface Builder in the XCode IDE. The reason for using the interface builder GUI tool rather than developing the user interface programmatically is that it creates a storyboard of how the containers interact with each other as seen in Figure 4.1.2 This a valuable tool to be able to show the client as the flow of the program can be understood without knowledge of the source code. This allows the client to request interface changes and see the effect of the change before actually having to compile and run the code for each change, saving time in development and getting quality feedback during demonstrations.

Storyboards were created on paper from the user stories during the design phase so the process was quite simple and fast as XCode provides a simple interface to drag and drop components similar to those drawn in the user story created storyboards.

### Figure 4.1.2

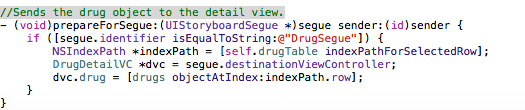


The main challenge of the user interface construction was getting it to resize dynamically for different screen sizes that are available for IOS. This is done by allowing all the objects to resize with the screen size but adding constraints to each of the objects inside the container to make the UI the. An example of a constraint is to make sure an object is always has a distance of 5 pixels to the right and of the container no matter how much they are stretched. This will allow for example a text area to resize to match the screen size to but always keep it centred. This is challenging because constraints can interact and override each other. When dealing with many constraints this can get very confusing into why a particular object is not behaving as it should when it is resized. XCode provides a feature “Resolve Auto Layout Issues” to help resolve this where it will fill in all the constraints that are needed to make the look like it does on the storyboard. Although this feature is very good at setting the positions the resizing is still an issue and needs other constraints such as keeping all the buttons on the calculator the same height and width ratio which have to be added manually.

The next step in the first iteration was to create the objects for the information on drugs and pathogens to be parsed to the UI. From the user stories decided upon in the initial meetings the objects could be created to suit the output displayed on the user interface. The main challenge of creating the objects was understanding the medical terminology into how each variable should be stored in the object. For example would the dose need to be a description or measurement (string or integer). Once a drug object and a pathogen object were created the view controllers could now be implemented.

### 4.1.3 Navigation

The objects are displayed in tables, drugs and pathogens each with a view controller to manage the operation of the user interface. The reason for using the built table components rather than use a customized UI component was to allow users familiar with IOS to be able to intuitively use the navigation of the app. The main view controllers are handled by a tab Each cell of the table links to a detailed view of the object displaying all the information about a particular drug or pathogen. This required the method of parsing data between view controllers based on what was selected in the previous one. IOS uses a method called UIStoryboardSegue.



This method looks for a segue created in the Storyboard then parses the object selected in the table to the detail view controller to be displayed. This also creates simple navigation for the user as to be able to use a segue the view controller has to be embedded into a navigation controller which automatically creates navigation buttons for the user to go back to the table as seen at the top of **A.1.2**.

### 4.1.4 Evaluation of First Iteration

To evaluate the mobile application a demonstration was given to the client and supervisor of the project. To do this a drug and pathogen object were created filled with test data that just showed what each field would represent. Three drugs and three pathogens were filled with data and displayed on the app. The client was then walked through each stage of the application and shown how the navigation worked and how it was similar to the low fidelity prototype. This was done to get feedback on the amount of data displayed, the look and feel of the user interface and if the application was useful at its present state.

The results from the demonstration were positive but some changes were needed for the next iteration. One suggestion was with the user interface for Drug Detail View as seen in **A.1.2** where both adult and paediatric are shown highlighted, the client suggested that it wasn’t obvious when you had selected the adult or paediatric information. This is critical because if a doctor prescribes an adult dose to a paediatric the drug can cause harm to the patient. So this would need to be changed for the next iteration such as making of the routes of administration bar smaller and making the adult and paediatric buttons larger and bordered.

Another criticism was with the fields displayed on the Drug Detail View as seen in **A.1.2**. There were some titles that needed to be changed to make the application more suited to the medical field such as “Treatment for” needed to be changed to “Indications”. The ordering was also discussed as doctors look at some fields more often than others so they need to be closer to the top. The field for the dose was misunderstood during the design phase and was thought to be just a number and a unit but a does also needs a description thus needing its own text area. A feature was suggested to be able to see only the pathogens associated with each drug that could be added in a later iteration.

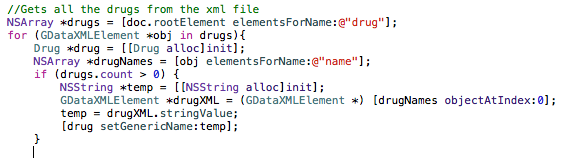
The Pathogen Detail View **A.1.4** was also discussed. It was originally agreed that the pathogen object was going to have a list of first line and a list of second line drugs. But the client decided this was not the best route to take upon seeing it on the user interface as it was feared that it may cause arguments among doctors who would argue some drugs are better than others. So for the next iteration will only show first line drugs would be shown but would include all of the drugs that were affective on the particular pathogen.

The simple calculator was tested during the demonstration to see if was easy to use and gave the correct answers. The client was asked to enter some simple calculations that would be normally used when calculating doses. It was shown to work well just to do some simple calculations without having to leave the app or get a calculator out. The interface was liked and no further changes would be needed to the calculator.

In conclusion to the evaluation there were some changes needed but application was highly praised due to the simple navigation and there were no major changes needed. The feedback helped refine the tasks for the next iteration to make sure the user interface was simple and easy to use. The feedback also helped clarify how the application was going to be used and what information needed to be displayed. As not all the information was needed at a glance the user interface could be made bigger allowing more information in each text field to give the user more detail about the drug or pathogen.

## 4.2 Iteration 2

The second iteration’s main goal was to implement server communication with the mobile application and create persistent storage for the mobile application. To do this the mobile application had to be able to access a URL get an XML file and convert the file to the drug and pathogen objects. The server would have to respond to this GET request and return a XML file.

Reading an XML document and converting it into an object was the main challenge of this iteration. On IOS a small part of the GData API was used to help implement this feature. Two files were used “GDataXMLNode.h” and “GDataXMLNode.m” which include many useful methods to find elements in the XML file and parse the data held in those elements to variables. 

The above code shows how parts of the drug object are read from the XML and stored into an array of drug objects. The GData library finds all the elements in the document object model that match the string and stores all the occurrences of the element in an array. To set all the objects all the root elements are found and stored in an array. For this segment of code “drug” is the root element, the code above finds all the drugs stored in the XML file then for each drug the child elements are found and the string returned that each element holds. Each string can then set the corresponding part of the object for example the name will set the name of the drug object, although some elements contain more than one string meaning arrays of strings were stored. This quite a challenge due to the complexity of the DOM. The drug DOM has many nested elements because of different information stored about adult and paediatric patients.

The need for the XML file to be so complex is due to trying to minimize all the communication with the server to make sure this mobile application can be used offline. Thus it can to be stored in one file that can be sent quickly from the server and stored on the mobile application.

The persistent storage of the XML file once received from the server was another challenge as the as it took time to familiarise with how GData extracted the data from a file. A problem occurred when trying to read from a recently saved file as GData can only process files that are saved in the main bundle of the mobile application as this is the only way to explicitly cast what type the file is. The main bundle is where read only files are stored not just the local directory of the mobile application. The way that this was solved was creating a method in which returns the file path of the main bundle. Once the file was created it could be copied to the main bundle and GData can understand what type of file it is.

///////////////here

Create simple REST server to send a small XML document with dummy data.

Object to XML JAXB

Construct object on server create server.

Client Review.

## 4.3 Iteration 3

Created database MySql,

It had originally been thought that the data for the drugs would be extracted from an already existing database that had been created for the clients website. The database was created using and web hosting service and when the client exported the SQL file it would not open in MySQL or WAMP using various SQL formats. It would open in a text editor but the syntax was confusing and the data was in many tables with various different IDs so would of took much time to find and organise the data extraction manually. This problem was unforeseen in the schedule and time was lost trying to create a solution to use this data.

The Decision was made to create a completely new database rather than using an existing one and editing it to fit the application. This however meant that the schedule had to be changed to allow for design and creation of database. There was an extra two weeks set aside for problems during development so this extra time was used for database design and creation. The design was first done using tables on paper then the tables were created using MySQL’s GUI for Visual Studio. This allowed the tables to be created as they were on paper and allowed for the relationships to be created visually. The SQL then was automatically generated to run on the MySQL server. The main challenge in the database design was getting the relationships to work correctly. The database should be orientated around the drugs table so when a drug is deleted that drugs information is cleared from all the tables in the database. //////HEEEEEEEEEEEEERRREE

Connect database to REST service JDBC database to the object

Return HTML of database

## 4.4 Iteration 4

Admin User Interface

Add/ delete from database

Advanced app features comparing drugs and finding pathogens.

# Evaluation

## .1 Results From User Testing

## .2 Future Work

Add more features to rest server, edit for example. Add different types of drug search feature.

Android.

## .3 Personal Reflection

Report is very hard.

Problems with clients database

Coding went well little problems. Research more frame works instead of reinventing the wheel.

Overestimated the amount of work that could get done in a the time three months sounds like a lot but it really isn’t.

## .4 Client Reflection

Was he satisfied ?

What does he wish happened?

Anything that went badly?

Professionally Managed?

Correspondance

Meeting Notes.

## .5 Dicussion on REST vs the Normal Way vs SOAP

# List of References

# Bibliography

Android. (2015, January 1). *Android, the world's most popular mobile platform*. Retrieved April 13, 2015 from Android Developers: http://developer.android.com/about/index.html

Apple. (2015, April 9). *About Itunes Connect*. Retrieved April 14, 2015 from IOS Developer Library: https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect\_Guide/Chapters/About.html

Apple. (2015, January 1). *XCode Features*. Retrieved April 14, 2015 from Apple Developer: https://developer.apple.com/xcode/features/

Bott, E. (2014, January 5). *Net Market Share vs. StatCounter: Whose online measurements can you trust?* Retrieved April 13, 2015 from ZdNet: http://www.zdnet.com/article/net-market-share-vs-statcounter-whose-online-measurements-can-you-trust/

Eid, A. (2014, June 13). *Apple's Swift is Great, but Objective-C is not Going Anywhere*. Retrieved April 14, 2015 from Huffington Post: http://www.huffingtonpost.com/ahmed-eid/apples-swift-is-great-but\_b\_5492239.html?

Elkstein, D. M. (n.d.). *Learn REST: A Tutorial*. Retrieved March 13, 2015 from rest.elkstein.org: http://rest.elkstein.org/

Google. (2015, January 1). *Get Started with Publishing.* Retrieved April 13, 2015 from Android Developers: http://developer.android.com/distribute/googleplay/start.html

Google. (2015, January 4). *Supported Devices.* Retrieved April 13, 2015 from Google Play Support: https://storage.googleapis.com/support-kms-prod/FBD905D8F0C3B7466021873B4B17A7BA6C7F

IMB. (2009, September 24). *Build a RESTful Web service using Jersey and Apache Tomcat*. Retrieved April 15, 2015 from IMB: http://www.ibm.com/developerworks/library/wa-aj-tomcat/

Mills, E. (2012, February 2). *Google now scanning Android apps for malware.* Retrieved April 13, 2015 from CNET: http://www.cnet.com/news/google-now-scanning-android-apps-for-malware/

Net Market Share. (2015, January 1). *Market Share Reports.* Retrieved April 13, 2015 from Net Market Share: http://marketshare.hitslink.com/

Quotient Integrated Solutions. (2014, August 28). *Benefits of Agile: Increased Customer Collaboration*. Retrieved April 9, 2015 from Quotient: http://quotient.net/

Shiny Development. (2015, January 1). *Average App Store Review Times*. Retrieved April 14, 2015 from Average App Store Review Times: http://appreviewtimes.com/

StatsCounter. (2015, January 1). *StatsCounter Global Stats For GB.* Retrieved April 13, 2015 from StatsCounter Global Stats: http://gs.statcounter.com/#mobile+tablet-os-GB-yearly-2015-2015-bar

StatsCounter. (2015, January 1). *StatsCounter Global Stats World Wide.* Retrieved April 13, 2015 from StasCounter Global Stats: http://gs.statcounter.com/#mobile+tablet-os-ww-yearly-2015-2015-bar

The Agile Alliance . (2015, January 1). *The Twelve Principles of Agile Software*. Retrieved April 3, 2015 from Agile Alliance: http://www.agilealliance.org/

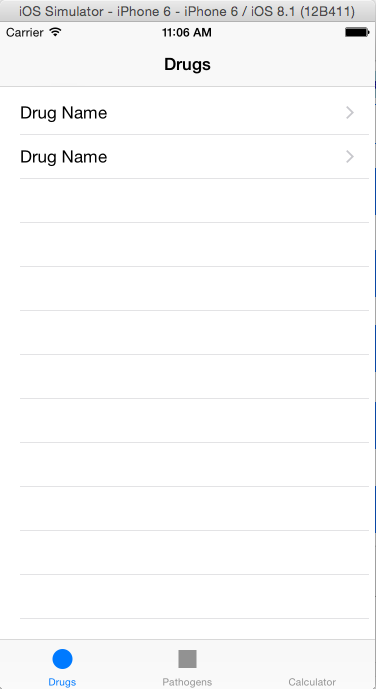
Vogel, L. (2014, August 20). *REST With Java (JAX-RS) Using Jersey - Tutorial*. Retrieved March 13, 2015 from www.Vogella.com: http://www.vogella.com/tutorials/REST/article.html

(If any.)

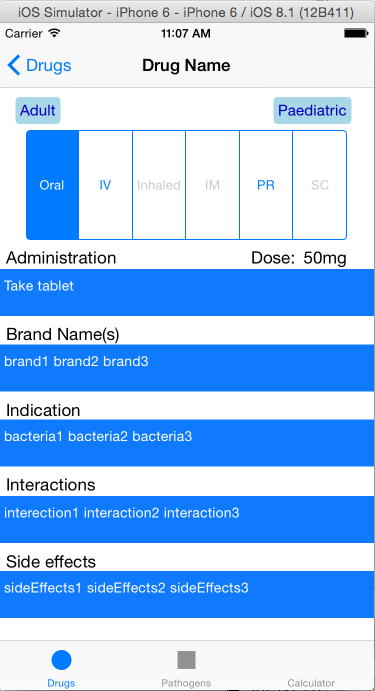
# Appendix A External Materials

## A.1 First Iteration User Interface

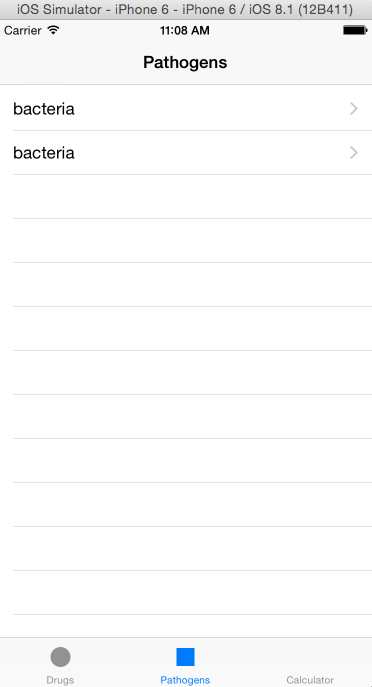
### A.1.1 Drug Table View



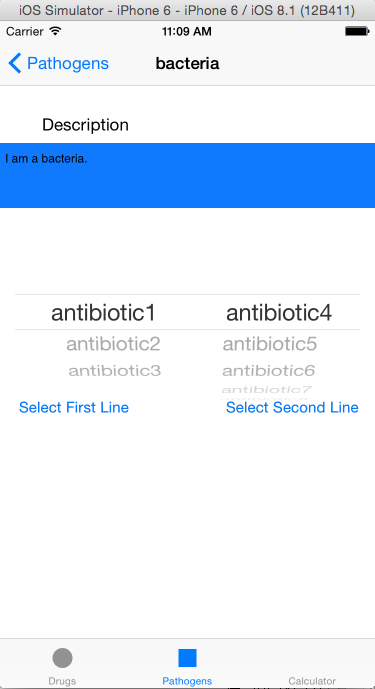
### A.1.2 Drug Detail View



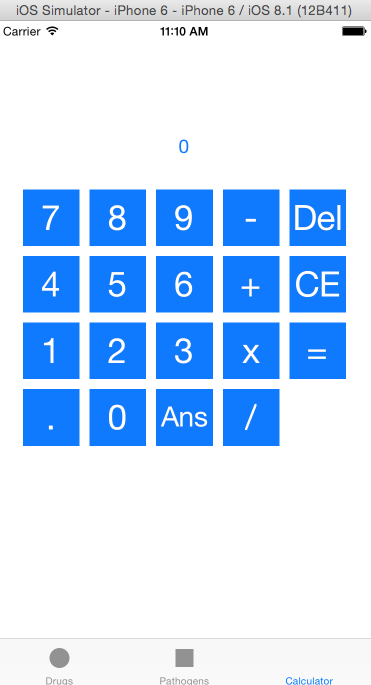
### A.1.3 Pathogen Table View



### A.1.4 Pathogen Detail View



### A.1.5 Calculator View



#### A.1.1.1 Level 4 Heading with ‘heading 4’ Style Applied by Pressing Ctrl Shift 4

Text under level 4 heading. Text under level 4 heading. Text under level 4 heading. Text under level 4 heading.

# Appendix B Ethical Issues Addressed

## B.1 Level 2 Heading

1. [↑](#footnote-ref-1)