

|  |
| --- |
| 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

## 1.5 Schedule

# 2 Background Research

## 2.1 Mobile Operating Systems

Android Blackberry IOS Windows

## 2.2 Programming Language For Development

Java Python C++/C Objective C Swift HTML JavaScript SQL

### 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( **[Ahmed Eid](http://www.huffingtonpost.com/ahmed-eid/)** June 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 Frameworks

Jersey vs Spring vs Django

## 2.4 XML Conversion Libraries

Gdata vs RestIOS JAXB etc

## 2.5 Integrated Development Environment

### 2.5.1 IOS

The chosen integrated development environment 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).

### 2.5.2 Java

Eclipse

## 2.6 Database Design

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

## 

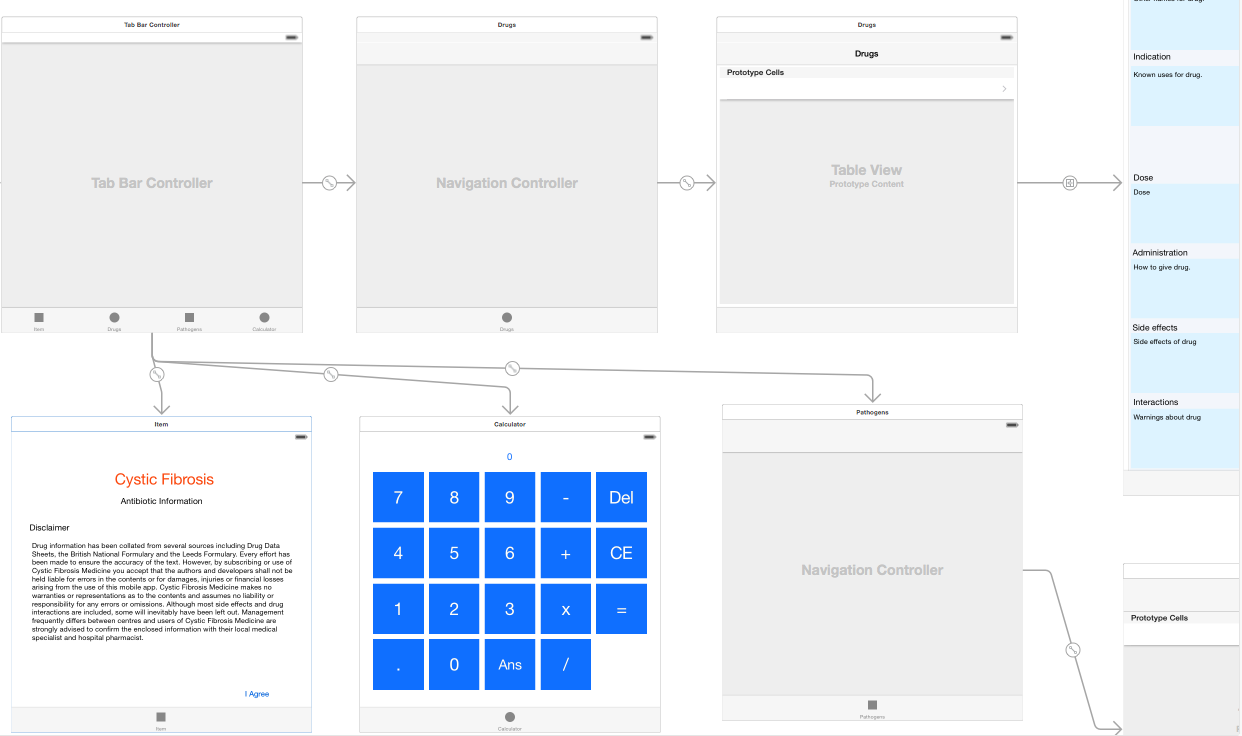
# 4 Implementation

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.

## 4.1 Iteration 1

Create working prototype from low fidelity prototypeCreate objectsLink user interface to objectsFill with temp dataClient Review

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 1. 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 change, compile and run the code for each change, saving time in development.



Figure

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 centered. 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 requirements gathered from the meetings a basic structure was drawn up for what con

## 4.2 Iteration 2

Make changes to interface per clients request

Convert from XML GDATA to object on mobile app.

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,

Problems with clients database not standard format caused time lost.

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

(If any.)

Apple Developer 2015 <https://developer.apple.com/xcode/features/>

[**Ahmed Eid**](http://www.huffingtonpost.com/ahmed-eid/) June 2014 http://www.huffingtonpost.com/ahmed-eid/apples-swift-is-great-but\_b\_5492239.html?

# Appendix A External Materials

<Level 1 Heading with ‘heading 1’ Style Applied by Pressing Ctrl Shift 1> Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading.

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

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

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

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

#### 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

Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading.

## B.1 Level 2 Heading

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

1. Footnote inserted using Insert Footnote on the References tab. [↑](#footnote-ref-1)