# Background, Analysis & Process

## Project Overview

The NHS owns a database containing monographs 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 requested Aberystwyth University via the Software Alliance Wales scheme 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 API URLs.

The aim of this project was to fulfil that request by creating a well designed, functioning and thoroughly tested Android mobile application. The completed application had to query the data to allow user 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 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 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 most accurate information is displayed to the user.

## Background and Analysis

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

## .1. Medusa website

The Medusa 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 for the selected drug. If calculator information is available then a button to open the calculator is also displayed.

The Medusa website 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 does not work effectively on mobile devices the only method of accessing the monographs in a portable manner is to print the monographs, this not an optimal solution as the printed information may also become outdated and the user will not know; this method is also not environmentally friendly.

Using the Medusa website allowed me to see how a monograph 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. A web application 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, a web application was not a possible solution.

Phonegap is a mobile development framework that allows developers to write mobile applications using HTML5, JavaScript and CSS3 instead of device specific languages. Phonegap then compiles the application written using web technologies into a hybrid application for multiple devices. A hybrid application is an application that appears to be a native application to the user but is not a native application 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 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; 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 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 and iOS applications would have been suitable for the project; they both have the ability to download the data over HTTP, parse the XML files, execute tasks in the background and save data for offline use within local databases.

Applications for iOS devices are created in Xcode IDE and are written in Objective-C, which is an object-orientated language based on C and C++. 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 which is an object-orientated language using either Eclipse or Android Studio 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 [1]. 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 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 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 and begin building applications. I also read the Android design guidelines, 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 I have executed there is currently no other mobile application that completes all goals of this project, but there are many libraries, frameworks and tools that I used throughout the project.

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

The Android SDK allowed me to create a native Android application that has the ability to utilise all hardware on the device. The Android SDK also provided me with the base UI framework, allowing me to create an application that an Android user can instinctively use.

Robospice 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. I used the Robospice library 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 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 I used whilst completing this project was StackOverflow. StackOverflow is an online community where users post programming related issues and the community help solve these issues. I used StackOverflow when I encountered issues 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 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 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.

## 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 application, as this approach would allow the application to be used by the most users.

**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 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. 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 my 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.

|  |  |
| --- | --- |
| **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 |

## 1.3.3. 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 more efficient a 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 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.

## 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.

### Methodology

Initially an eXetreme programming methodology adapted for solo programming was the chosen methodology for this project. Using an eXetreme 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, that that 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 eXetreme 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 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 easy to create errors.

After realising that eXetreme programming was not the best methodology for this project I decided to move the project to use the waterfall model, whilst still using features from eXetreme programing that I believed would enhance the project. Using the waterfall model 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 then 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.

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 eXetreme 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.

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 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 eXetreme 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 eXetreme 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 eXetreme programming 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 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.

### Planning

### Planning is vital to the success of a project and was inherently brought to the project through the waterfall method. 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.

### Prototypes

Before starting the design stage of the waterfall model 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 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 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.

## 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 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 as a backup for the repository. Should the developers of the NHS want to see the process I used to create they application I can share the repository along will 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.