# 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.

## 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. 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. The feedback left by the NHS confirmed that the final application is 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.

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

## 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.

## 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.

## 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 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 over AsyncTasks. 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 add 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.

## Methodology

The methodology used was adapted twice throughout the project. Originally a pure eXetreme programming was planned, and then the methodology was changed to the waterfall model with test-driven development 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 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 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.

## 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 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 style comments, which has been compiled into a HTML document, allowing future developers to gain knowledge of the system easily.

The implementation allows a large amount of the applications to be modified by editing only XML files. 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.

## 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 the all downloads to start simultaneously and using multiple threads. As the download progress (Download x drugs of y) requires the drug index to be download, a higher priority should be set of 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. 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 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.

## 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.

**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.

**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.

**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.

**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.

**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.

**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.

**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 doctors surgery.