Liverpool John Moores University

Department of Computer Science
Final Year Project

Project Title: Delivering health passports using location based QR codes.

Student Name: Matthew Nethercott

Student ID: 718741

Programme name: Computer Science, Software Development Project (Scheme 1)

Date: 26th April 2019

This project is submitted for the module 6001PROJ/6000PROJ and complies with all relevant LJMU academic regulations, including plagiarism and collusion.

Abstract

Although legislation, such as, the Equality Act (2010) has sought to aid in the removal of discrimination faced by those living with learning disabilities, it has not been as successful as hoped, in healthcare. Healthcare professionals lack the correct training to effectively care and treat those living with learning disabilities. Parents and carers are there to help healthcare professionals to care and treat for their patients, however, they are often ignored with regard to clinical and medical information.

Behaviour difficulties seen in those living with learning difficulties are often misconceived as a consequence of the condition rather than symptoms of the medical difficulties they are facing.

Health passports have often, in the past, been used to inform and educate healthcare professionals to better treat their patients, however, they are rarely updated and use outdated means of communication.

This paper follows the Agile development of an eHealth Passport. This passport will allow for clinical data to be stored on both an App Server and a cross platform Mobile Application. The proposed cross platform mobile application uses location services, QR codes, and notifications to provide non-invasive access to a user's health passport for health professionals.

Acknowledgements

A special thanks to Tom Dawson and Rescon for their continuing and ongoing support during this project.

Thank you to Professor Abdennour El-Rhalibi for his guidance during the project.

Finally, for the continuous support a special thanks to family and friends.

List of Contents

List of Tables

List of Figures

Introduction

Currently, health passports use traditional paper-based methods of delivery, however, this, always requires users to carry a paper-based passport on them. These passports range in size but passports of lengths up to 36 A4 pages are frequently used (Record, Family and Version, n.d.). This project sees the creation of a Cross-Platform (iOS and Android) Mobile Application. It is hoped this project can be used to replace the current traditional paper-based models. Patients typically always have their phone with them, so recreating a Health Passport as a mobile application provides the perfect medium to store and hold their passport. However, although this solution is ideal for holding a Health Passport, many mobile phones are locked with a pin code, a passcode, or a biometric restricting outside access to the passport.

In order to overcome the issues of the security features on a mobile phone, this project sees the creation of a health passport, which with the use of a QR code, will activate and display to a user when they enter a healthcare facility. Location Services, specifically GPS, can be used to derive the device's location. Using reverse geocoding and partial string matching, it is possible to accurately determine whether a device is located within a certain building by analysing the postal address.

Barcodes and QR codes can be used to create a unique point of access to a user's health passport. Generating a QR code by using a health passport allows for a lightweight and easy to access eHealth passport to be created. By combining the QR code with a Geolocator, QR codes can be activated when the geolocator detects the device is within a certain building or area. Partial string matching can be used to process the address for key phrases, addresses, and areas. Once it has been determined device is in a supported location, notifications can be used to display the passport's QR code on a lock screen, which provides easy, restricted access for someone to see complete medical information on a user. This can be used to accurately and effectively treat the owner of the health passport, regardless of the situation of their health.

This document sees the creation of the eHealth Passport, it is structured in the following way to define and outline the full creation: background, literature review, requirements, design, implementation and testing, evaluation, project planning and conclusion.

Background

It is a known fact, people living with learning disabilities often have decreased levels of health (Emerson and Baines, 2011). Learning Disabilities are often defined as lifelong conditions, for which there is no cure, that cause a significant intellectual disability (Brown et al., 2010, Cited in Bradbury, Jones et al., 2011). However, few are diagnosed with a learning disability at birth (Alborz, Mcnally, and Glendinning, 2005). It is a condition often diagnosed during school where intellectual ability differs from normative levels (Alborz, Mcnally and Glendinning, 2005). Examples of learning disabilities include (but are not limited to): Down's Syndrome, Autism, and William's Syndrome (MENCAP, n.d.). In 2004, it was found 2 per cent of people who were registered on a general practice register are living with a learning disability (NHS National Patient Safety Agency, 2004). Which accounts to approximately 1.3per cent of the general population (Alborz, Mcnally, and Glendinning, 2005). It is estimated, 26 per cent of these people will be in hospital every year (MENCAP, 1998, Cited in Vaz, 2010). With an increased risk of being admitted to hospital (general population is 14 per cent likely to be admitted (National Patient Safety Agency, 2004) and specific needs to be taken into consideration, it is imperative for accurate arrangements to be made to effectively treat and care for those with learning disabilities.

Learning disabilities, however, affect a patient's ability to recognise and communicate their health needs and conditions (Alborz, Mcnally, and Glendinning, 2005) meaning they are unable to effectively communicate to a healthcare professional. Due to this, it is required for healthcare professionals to recognise the needs, health condition, and any adjustments they need to be made in order to effectively treat the patient. However, often, professionals lack training and the necessary skills needed to identify these needs in patients with learning disabilities (MENCAP 2007, Cited in Vaz, 2010). This often leads to patients with learning disabilities being denied basic needs and support during visits to hospitals.

This lack of training causes incorrect discharge arrangements, errors with medication, and poor continued care, where patients see different doctors each time causing repetition of previous issues (Ali et al, 2013). Accident and Emergency departments (A&E) are often expected to make reasonable adjustments to assist with the care of patients. However, often zero or incorrect adjustments are made (Morris, 2018) and additional support is not provided to patients with learning disabilities (Ali et al, 2013). April 5th, 2011 saw the introduction of the Public Sector Equality Duty (a subsection of the Equality Act 2010). This act forces public authorities to "eliminate discrimination, harassment, victimisation and any other conduct that is prohibited by or under this Act" (Art 2010, p96). However, as discussed, previously, this has not been the case and consequently, causes a rise in symptoms of anxiety, for the patient (Vaz, 2010). Specifically, for patients living with Autism Spectrum Disorder, A&E departments already cause a rise in anxious feelings due to the change of daily routine, clinical environment and increase in sensory engagement (Vaz, 2010)

Parents and carers provide a vital role in the treatment of patients with existing learning disabilities. They know their child's personal sensitivities, medical histories, and how to best communicate with them (Vaz, 2010). However, parents and carers

are often ignored with regard to the medical treatment for their children living with Learning Disabilities. There have been cases of doctors refusing to investigate patients due to misunderstanding behaviour difficulties are attributed to their learning disability (Ali et al, 2013). Cases of this often result in further complications and serious misdiagnosis, with the most serious cases potentially leading to permanent irreversible damage (Ali et al, 2013).

However, although a valuable asset, parents and carers are not always available or reachable to assist with the treatment of their child. For example, if someone was to collapse in the street with no witnesses and is unconscious, how are healthcare professionals to treat the patient with no information about the patient? Apple and Android devices do provide access to an 'Emergency' feature where patients are able to provide basic information required by paramedics. However, these are not mandatory to fill out and often are not filled out or missed by a paramedic. These emergency features often only allow for basic information to be captured like height, weight, blood type, etc. This leaves only one option for gaining information about the patient; unlocking their phone, which professionals often do, in the circumstance of the patient being unconscious. A professional unlocking the phone to extract information is a privacy, ethical, and potentially legal issue that is being faced by healthcare professionals. Although the Computer Misuse Act (CMA (1990) predates smartphones and modern computers, its terminology is still relevant to the use of these devices. The CMA defines unauthorised access of a computer system with 3 factors:

- "he causes a computer to perform any function with intent to secure access to any program or data held in any computer" (HMMO 1990, p1)
- "the access he intends to secure is unauthorised;" (HMMO 1990, p1)
- "he knows at the time when he causes the computer to perform the function that that is the case." (HMMO 1990, p1)

Therefore, under the CMA it can be seen, unlocking a patient's device by the use of the biometric password, PIN, or traditional passcode is against the CMA.

One way to help tackle this issue is with the creation of Health Passports. Patients with Learning Disabilities have been encouraged to create a Health Passport, which provides healthcare professionals an insight into a patient's current health situation whilst breaking the barriers of communication issues faced by patients. The Health Passport model is a favoured model by both Parents/Carers and Healthcare professionals due to its quick and easy nature to fill out and read (Lee, 2012). They are designed to provide an overview of the patient's health and are designed to be filled out as accurately as possible, however, do not need to be completed in full (Lee, 2012). They are typically used to help provide information at the point of contact with the patient and help to guide healthcare professionals to effectively treat and care for the patient. They are widely recognised and used throughout the United Kingdom by people living with Learning Disabilities (Lee, 2012), however, they are often paper based so may not always be on the patient when they are admitted for an emergency.

However, current technologies allow for a non-invasive method of attaining information typically kept within a Health Passport from a mobile device. Non-invasive is defined in this report by a method in which the patient's privacy has not

been violated in order to gain the information. In 2015, 68 per cent of Adults in the UK own a smartphone with a staggering increase to 91 per cent when consulting the demographic ages of between 18-34 (Poushter, 2016). Therefore, it can be concluded, in the UK, most patients will be carrying a smartphone device with them when entering a healthcare facility during an emergency or regular appointment. Meaning it is possible to use the smartphone to display a non-invasive prompt to the healthcare professional from the patient's device to access their Health Passport.

Due to the nature of the data, it is required for the data to be kept secure. GDPR (General Data Protection Regulation), a data protection law for residents of the EU, states all personal data must be processed in a way in which keeps the data secure from unauthorised access (Ico.org.uk, n.d.). The use of a login system, which will allow the device to lock itself when navigated away, will allow for personal data provided by the user to be kept secure if someone unauthorised was to access the device.

Literature Review

Current System

There have been reports of health passports being used for many years; some as early as 2005 (Talkback, n.d). They are designed to help people living with learning disabilities to easily and effectively communicate with healthcare professionals (Lee, 2012). However, they are often outdated, paper based and without a consistent design. Although there is no consistent design, many have a similar structure with the same data required to fill out the passport. There are many different health passports used across the UK each provided by the respective Clinical Commissioning Group (CCG). This literature will review and evaluate the design of three existing health passports used within the UK.

Using the results of this literature review of the current system, requirements for the design and implementation of the eHealth passport will be created. This will consist of the clinical requirements for the project, which will see a focus placed on vital clinical information that is needed for a successful health passport. To get these requirements, health passports provided by East Kent Hospitals University Trust, The Royal College of General Practitioners, and the St Helens CCG will be reviewed and assessed. These health passports were chosen due to being widely publicly, and freely available online.

Table 1: represents what information you are able to provide in each of the different health passports. Each of them has the ability to allow a patient to provide similar information, however, at varying levels of detail and ease to complete.

		Next of Kin	Name	NHS Number	Carer	Personal Care	Address	Phone	Medical history	Medication	DOB	GP Details	Allergies	Communication Needs	Risk/Safety	Seeing and Hearing	Eating and Drinking	How to take medications	How to go to the bathroom	Moving ability	Managing with pain	How to comfort	Sleeping routine	Level of support required	Things that will make stay better	Things that will make stay worst	All about me	Tasks able to complete with suppor	What my carer wants you to know	Risks	Medical Condition
1	Royal College	X	X	X	X		X	X	X	X																	X	X	X	X	X
2	East Kent	X	X	X		X	X				X		X	X	X	X	X	X	X	X	X	X	X	X	X	X					
3	St Helens and Knowsley	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X

		Useful information	Notes	Date completed	Mental capacity assessment	Social Services number	Power of attorney	Spirituality	Disabilities and Impairments	Medications	Discharge arrangements	I am happy when	I am sad when	Dressing and undressing	Sitting	Standing	Mobility aid?	Brushing my teeth	Religion	Medical Conditions	Consent	Behaviour	Accessible version
	1	X	X																				
	2				X	X	X	X	X	X	X	X	X	X	X	X	X	X					
	3																		X		X	X	X

St Helen's and Halton CCG provide the best user experience with their health passport. They layout the information using a traffic light model, where the indications in the traffic light correspond to the importance of the section of the health passport. This allows for information to be quickly interpreted faster than large blocks of text. Although it presents less information than East Kent, its presentation and the value of the information is better than that of East Kent. Royal College, however, gathers very simplistic data with a free text theme. Although it gathers the least amount of data, it requires the most space with 35 pages being used. This causes a need for the patient to carry 35 A4 pages for their health passport, which could become destroyed in their bag due to unforeseen factors.

Table 1 demonstrates how each of the health passports follows a consistent structure for what data is collected regarding the patient. For example, all 3 collect a base set of information about a patient, this includes but is not limited to: Name, DOB, Phone, and NHS Number. This information is often required by all of the health passports as it allows healthcare professionals to lookup more information on their systems.

Table 2 displays a weighted value for each field that is encountered in the three health passports. This weighted value was used to define the requirements used to create the eHealth Passport.

Table 2: Weighting of Elements from Health Passports to see how often they occur in passports

Element	Occurrence	Occurrence position
Next of Kin	4	1
Name	3	1
NHS Number	3	1
Carer	3	1
Personal Care	3	1
Address	2	1
Phone	2	1
Medical history	2	1
Medication	2	1
DOB	2	1
GP Details	2	1
Allergies	2	1
Communication Needs	2	1
Risk/Safety	2	1
Seeing and Hearing	2	1
Eating and Drinking	2	1
How to take medications	2	1
How to go to the bathroom	2	1
Moving ability	2	1

Managing with pain	2	1
How to comfort	2	1
Sleeping routine	2	1
Level of support required	2	1
Things that will make stay better	2	1
Things that will make stay worst	2	1
All about me	1	1
Tasks able to complete with support	1	1
What my carer wants you to know	1	1
Risks	1	1
Medical Condition	1	1
Useful information	1	1
Notes	1	1
Date completed	1	1
Mental capacity assessment	1	1
Social Services number	1	1
Power of attorney	1	1
Spirituality	1	1
Disabilities and Impairments	1	1
Medications	1	1
Discharge arrangements	1	1
I am happy when	1	1
I am sad when	1	1
Dressing and undressing	1	1
Sitting	1	1
Standing	1	1
Mobility aid?	1	1
Brushing my teeth	1	1
Bathing and washing	1	1
Religion	1	1
Medical Conditions	1	1
Consent	1	1
Behaviour	1	1
Accessible version	1	1

As seen in Table 2, all elements where given a weighting between 1 and 10 (however, 4 was the highest seen). This weighting was then highlighted in green – the greater the occurrence the stronger the shade of green that was applied to the cell. Every element was seen to have a base value of 1 as each element appeared at least 1 time to appear on the list. The table was then filtered for this example with

the final column. Only the first occurrence of each element was displayed in this table to keep the size of the table down.

Using Table 2, the requirements specification for the proposed system was created. Elements that have an occurrence value of greater than 2, was deemed necessary to add to the requirements specification as it is needed to effectively treat patients. However, the higher the value, the higher the priority of it to be implemented. For example, Next of Kin appeared in the health passports four times against Sleeping routine which only appeared twice. This means Next of Kin will be required to be implemented before Sleeping routine as it is seen to have a higher importance in healthcare.

Software Methodology

In software development, there are three main methodologies that are used during development; Waterfall Model, Agile Development, and the V Model. These all have merits and downfalls. Some of the methodologies discussed suit better short development cycles and some support longer pre-defined cycles. A comparative study will be carried out to decide which is best suited to this project.

In this project, the requirement is to develop a Web Application, Mobile Application, and a MySQL Database. Currently this will be replacing a traditional paper-based system, which is be manually completed by hand and, carried by the user at all times. The new system will allow the user to fill out their health passport on a Mobile Application, store it on a centralised MySQL database, and have a read-only portal held on a web application for anyone to view, after authentication.

Waterfall Model

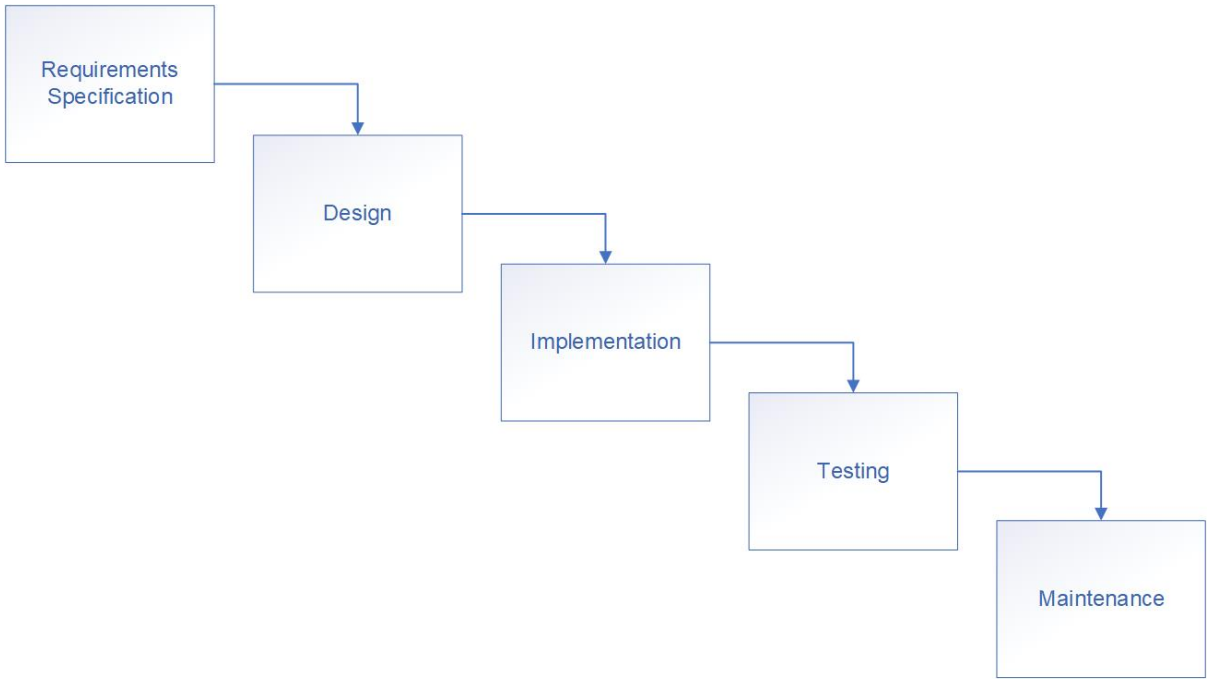


Figure 1: A typical waterfall model methodology demonstrating the strict nature of the model when used to manage a software development project (Balaji and Sundararajan Murugaiyan, 2012).

The waterfall model follows a sequential development process where each step of the lifecycle is completed before the next one starts. The waterfall model is ideal when the requirements are known from an early date in the development cycle and are unlikely to change during the development (Balaji and Sundararajan Murugaiyan, 2012). However, due to each stage being completed sequentially after the previous stage, testing is not started until the development phase is completely finished. This means any underlying issues found during the implementation phase may not be fixed until the next development iteration (Balaji and Sundararajan Murugaiyan, 2012). The original waterfall model design did allow for feedback to be given at each stage but, it has been adapted to be strictly linear (Pressman and Maxim, 2015). When an adaptation needs to be made for a piece of software, the requirements are well known before hand of what the adaptation is. This is an ideal scenario where the waterfall model should be used (Pressman and Maxim, 2015). The waterfall model has been around for over 40 years making it by far the oldest software methodology, however, due to its age many see it as outdated (Pressman and Maxim, 2015). Stakeholders and Customers are advised patience with this model as it is unlikely, they will see any of their product until late in the process. As seen during the analysis of the current system, the requirements for this project are already known due to the extensive amount of guidance for designing health passports in place. As previously stated, this allows for the Waterfall Model to be used to develop this project.

V-Model

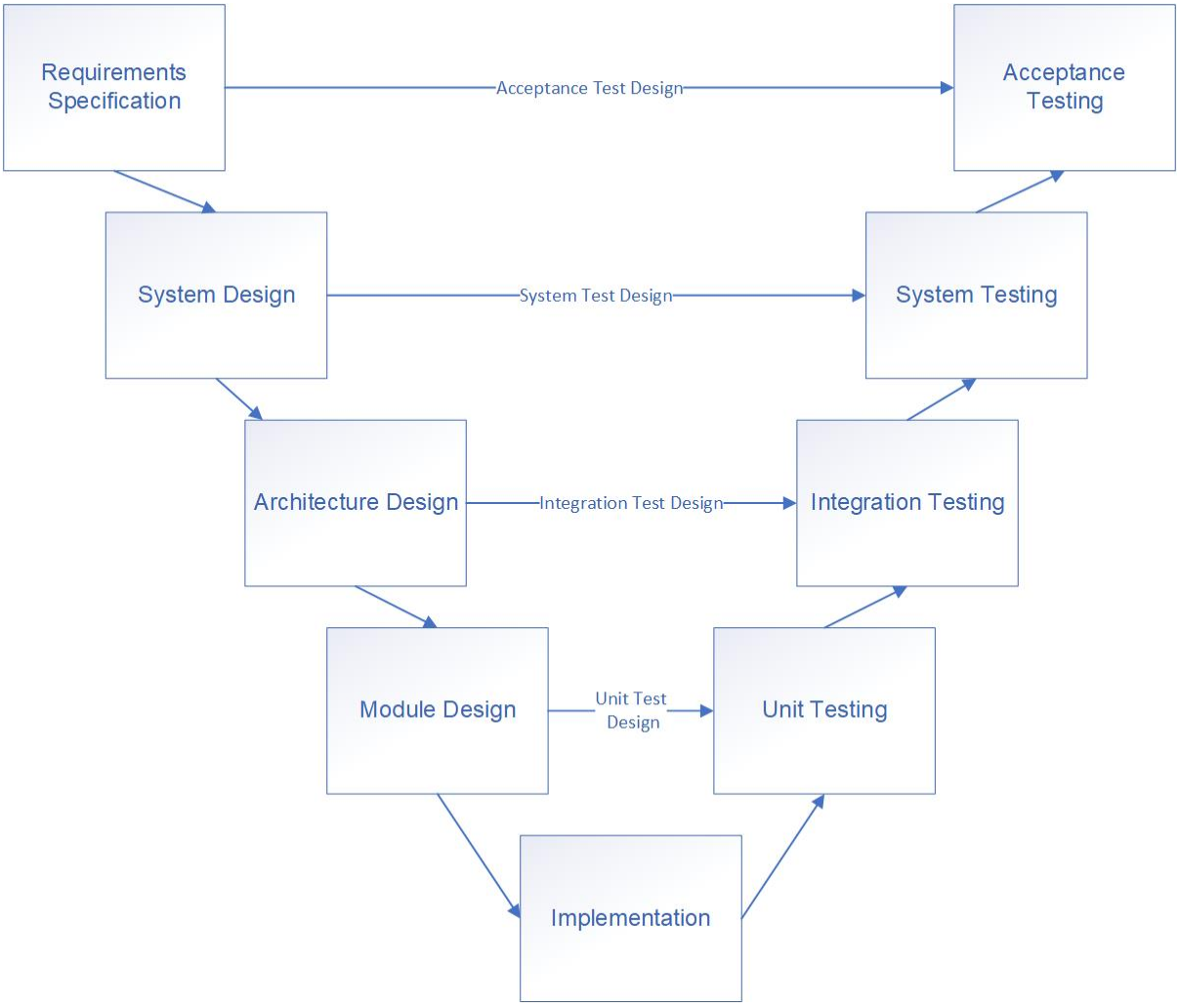


Figure 2: Typical workflow when using a V-Model methodology (Balaji and Sundararajan Murugaiyan, 2012).

The V-Model, often referred to as the Verification and Validation Model (Balaji and Sundararajan Murugaiyan, 2012), is built upon the waterfall model where there are actions in place to allow for feedback at each section allowing for there to be a small amount of iteration to get the development process correct at each stage. This allows for testers to be consulted when the software is being implemented to ensure software is not delivered in an unfinished state. This is ideal for projects that are expected to take a long time to develop due to the constant validation that needs to be performed after each step. It is the most rigid of all the software methodologies and if any of the requirements change, then the requirements documentation and test documentation needs to be altered to match (Balaji and Sundararajan Murugaiyan, 2012). For this project, this may not be ideal as the time-span in which the project is to be completed is very short.

Agile Development

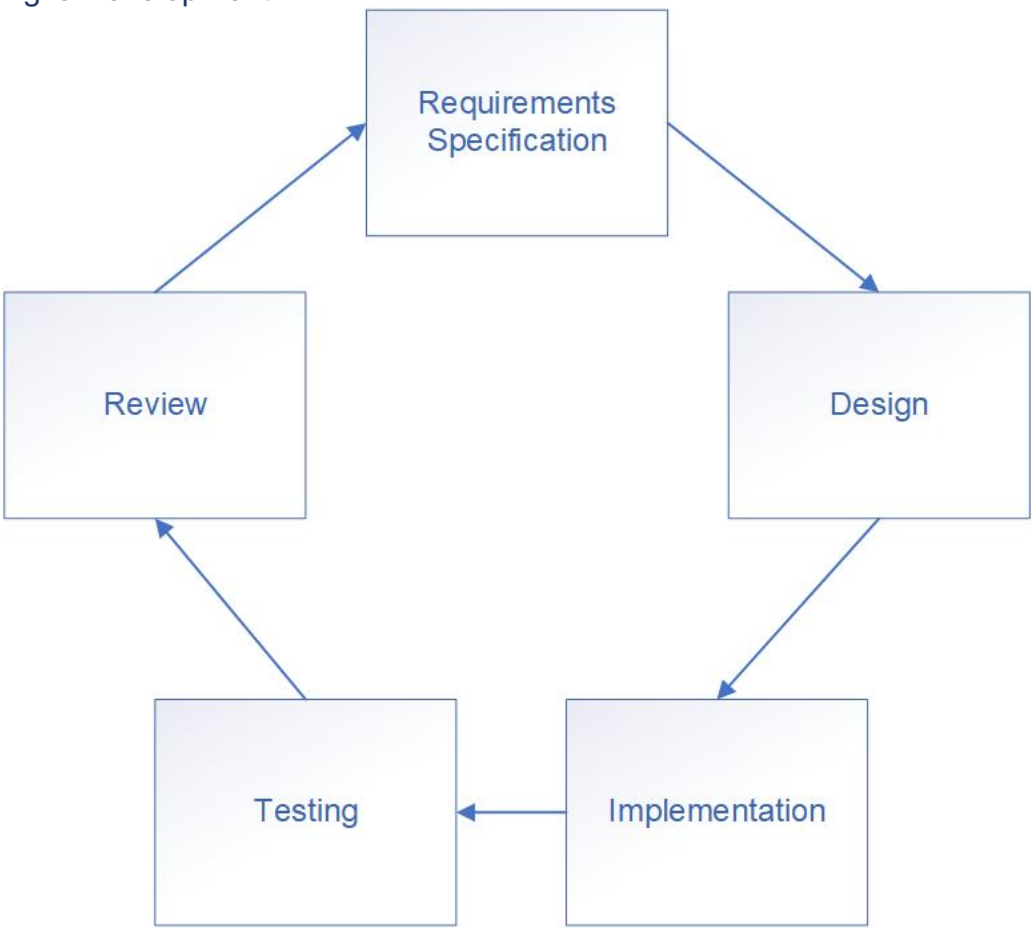


Figure 3: Typical workflow when using an Agile Development methodology to manage software development (Balaji and Sundararajan Murugaiyan, 2012).

Agile Development or the Moving, Quickly Model (Balaji and Sundararajan Murugaiyan, 2012) allows for quick and adaptive development of a system. It is ideal for when the requirements may not be fully known for the system or a short development cycle is required. During Agile Development, software is often created and delivered to customers within weeks rather than months (Balaji and Sundararajan Murugaiyan, 2012). Due to the iterative development seen during the Agile Development lifecycle, it is possible to respond quickly to changing requirements and still have a robust system created (Balaji and Sundararajan Murugaiyan, 2012). During Agile Development, it is often seen that the requirements

specification will take place before any development has been started. However, this does allow for the requirements to change and adapted during the development cycle. This can easily be achieved due to the constant feedback from the customers and stakeholders. Due to the monthly meetings required for this project, the Agile Development model is ideal as it will allow for monthly sprints to be performed, in which software is iteratively developed and feedback is gained to fine tune and refine the end product.

After analysing the above methodologies, it has been decided to use an Agile approach to develop the system. Although the requirements have already been defined, the short time span and constant feedback allow for an agile approach to be used. It is the hope that during each four-week period, software can be designed, implemented and tested to ensure progress is made at a steady pace to reach the deadline. An industry standard approach of combining four-week periods with planned Gantt charts to outline what tasks are to be implemented in each code sprint. These Gantt charts will be used to define what is to be designed, implemented, and tested during each four-week period and to keep the project on track for its deadline.

Requirements

Using the Literature Review of current commercially available health passports, the following requirements have been derived to create the eHealth Passport for this project:

Functional Requirements

- QR codes to appear on device lock screen when in healthcare centre (EG: Hospital),
- MySQL Database is used on Web Application,
- SQLite Database is embedded on the Mobile Application,
- Web Application is written in PHP,
- Mobile Application is written using Xamarin Forms (C#),
- Logged access is viewable by the owner of the health passport,
- When the user navigates away from the app, they are required to login again,
- Users are able to view a read-only view whenever they want,
- Anyone has the ability to scan the QR code without creating an account.
- QR Code directs the scanner to a web application.
- Data is secure during transmission and rest on App and Server.
- Users are able to provide their:
    - Next of Kin
    - Name
    - NHS Number
    - Carer
    - Personal Care
    - Address
    - Phone
    - Medical History
    - Medication
    - DOB
    - GP Details
    - Allergies
    - Communication Needs
    - Risk and Safety
    - Seeing and Hearing
    - Eating and Drinking
    - How to take Medications
    - How to go to the bathroom
    - Moving Ability
    - Managing with pain
    - How to comfort
    - Sleeping routine
    - Level of support required
    - Things that will make my stay better
    - Things that will make my stay worst
- Basic information is provided and logged when someone views a read only version of a health passport:
    - Address,

    - Forename,
    - Surname,
    - Date/Time
    - IP address
- Error logging to capture:
    - File
    - Directory
    - Line
    - Error Message

Non-Functional Requirements

- The location of the device is gathered at least every 30 minutes.
- Secure authentication is used to determine whether the QR code has actually been scanned.
- The location is to be gathered whilst not significantly affecting battery life
- Code is fully documented and commented
- Screenshots cannot be taken when the mobile app is in full view
- Full fault detection on App server to log any errors
- Ability to dynamically alter the keywords for geolocation string matching

Desirable Requirements

The Following requirements are desirable and to be implemented if there is time at the end of the development phase:

- Users are onboarded when they use the app for the first time so know what to record.
- User is able to print a PDF version of their eHealth Passport if they wish to have a paper-based copy.

Design

UML Use Case

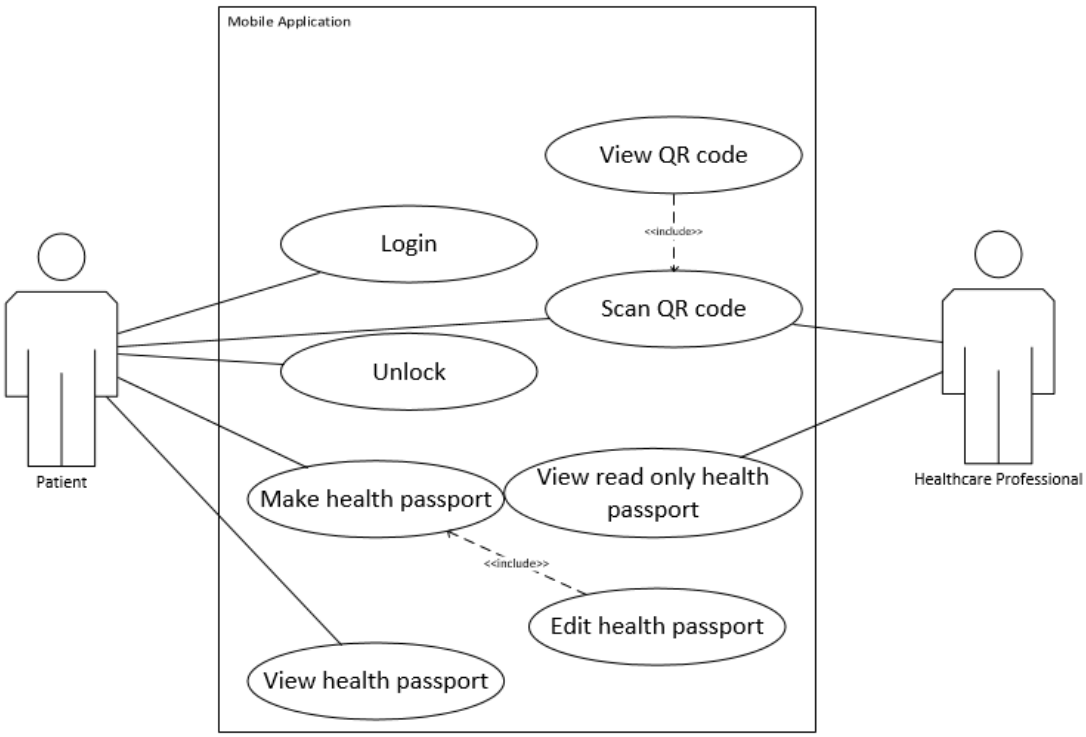


Figure 4: Use case diagram demonstrating the roles within the project with adequate access rights provided.

Figure 4 shows a use case diagram which shows the designs for both user types that are to use the system. The use case diagrams show a high-level of actions that they are able to perform on the system. For example, it shows how patients are able to manage their health passport which includes making one and editing it. A health care professional then has the ability to view the passport in a read only view however, they can never edit it. Unless it is their own passport.

Application Design

Xamarin Forms is to be used to develop the mobile application. Xamarin Forms provides a cross-platform library to create mobile applications utilising C# that contain the ability to run on iOS, Android and Universal Windows Platform (Docs.microsoft.com, n.d.). For this project, iOS and Android will both be targeted for the development of the mobile application. It will be required for both devices to have internet capabilities, location services enabled (with permissions granted), storage space available – to hold an embedded database. Finally push notifications will need to be enabled to gain the full potential of the application, however, a minimal version of the app will still be available. Location services and an Internet connection will always be required for the app, due to background tasks, so therefore, this will be required from the user and the device.

Google Play requires all new Apps created from August 1st 2018 to target at least Android 8.0 (API Level 26). The App will therefore be created to at least this Android version or higher for maintainability. Due to the mobile application using background updates for the location services aspect of the application, it is required for all iOS devices to be running iOS 9+. With this taken into consideration, the latest version of iOS will be used to develop the application however, a provision to run on iOS 9 will be made.

It will also be required to create a small web application and web service. This web application will be used to display a read only view of the health passport. It will be required for up to date browsers to be used by the end user. The majority of testing on this application will be performed on Google Chrome and Mozilla Firefox, however, the latest version of Internet Explorer should be considered due to its continued use in healthcare (No other versions will be provisioned due to its discontinuation).

Class design

The mobile application will be designed using an MVVM structure. MVVM is an architectural structure that separates the GUI code from the back-end logic. In a Xamarin Forms application, MVVM separates the code into view, model, and ViewModel which separates the XAML (design) code from the C# code by using Data bindings (Chang et al., 2015).

It will be required to separate each class type in the MVVM structure into separate packages so the code is easy to follow. MVVM provides easier development cycles, which consequently, create an easier codebase to maintain test and develop. The MVVM structure allows for code reuse, which lowers the amount of backend logic that is required for the mobile application. For example, backend logic for the Unlock and Login views can be reused by utilising one ViewModel, it will be required only to create two separate views to use this logic.

It is required to design either into the backend logic of the mobile application or as a separate web service, an integration with a global database to hold all user data. This will be separate to the embedded SQLite database that will hold just the currently logged in user's data. An investigation will be carried out during the section of this project specification to outline which will be created for this application.

Figure 5 shows the UML class diagram created for the cross-platform mobile application. As seen, seven packages are designed. Main, Views, Models, Templates, ViewModels, Services, and Security. Each of these packages has a different role within the mobile application and is required for the application to successfully run. The roles of each of the packages are as follows:

- **Main**: The main package for the mobile application. It contains the two main classes for the application with entry points for when the application starts running.
- **Views**: This package contains all the front-end code for the application. Each wireframe that is to be designed will be implemented in this package. Any direct backend code will be placed here too however, most code will be placed in ViewModels.

- **Models**: This is the package that deals with all data classes. Objects, such as the JSON that comes from the app server, will use Models to map the data so that the mobile application can handle it.
- **ViewModels**: This contains all functional back-end code with bindings for the front-end placed in Views. Each view will have a ViewModel created with some views utilising the same ViewModel to allow for reusable code to be created (Login and Lock views for example).
- **Templates**: Templates contain any templates for Views that can be used to create a reusable front-end code. The View will link in the template so that it is the same layout for each View that implements the code.
- **Services**: Services is the package used for any services that may be required by the ViewModels. Services such as integrating with the SQLite database or pulling data from the static DataStore. The static DataStore is instantiated on App load to all load data from the SQLite databse to minimise load times whilst using the mobile application.
- **Security:** is the package that deals with all encryption, and hashing performed on the mobile application.

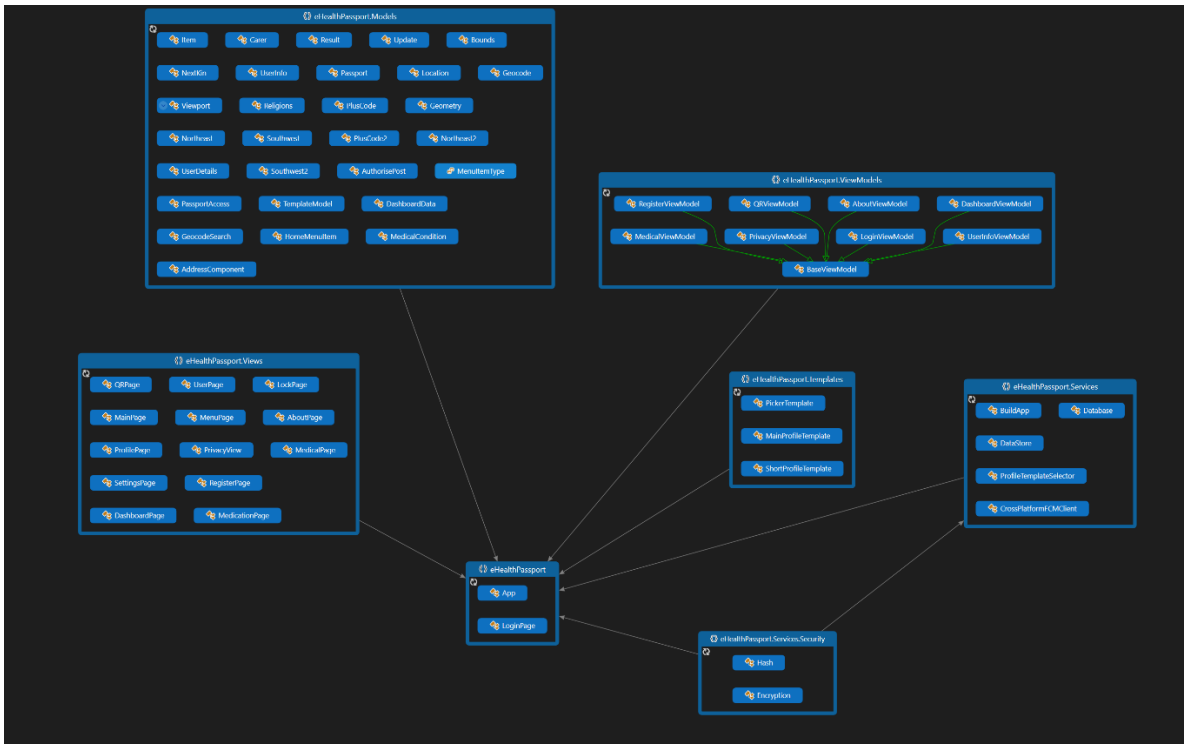


Figure 5: UML Class Diagram demonstrating the MVVM Structure

The diagram in figure 6 represents the main project used by Xamarin Forms to create the mobile application. In order for a native mobile application to be created on iOS and Android, three projects will be created. When targeting a specific device, Android for example, two of the projects will be used, the Main and an Android specific project.

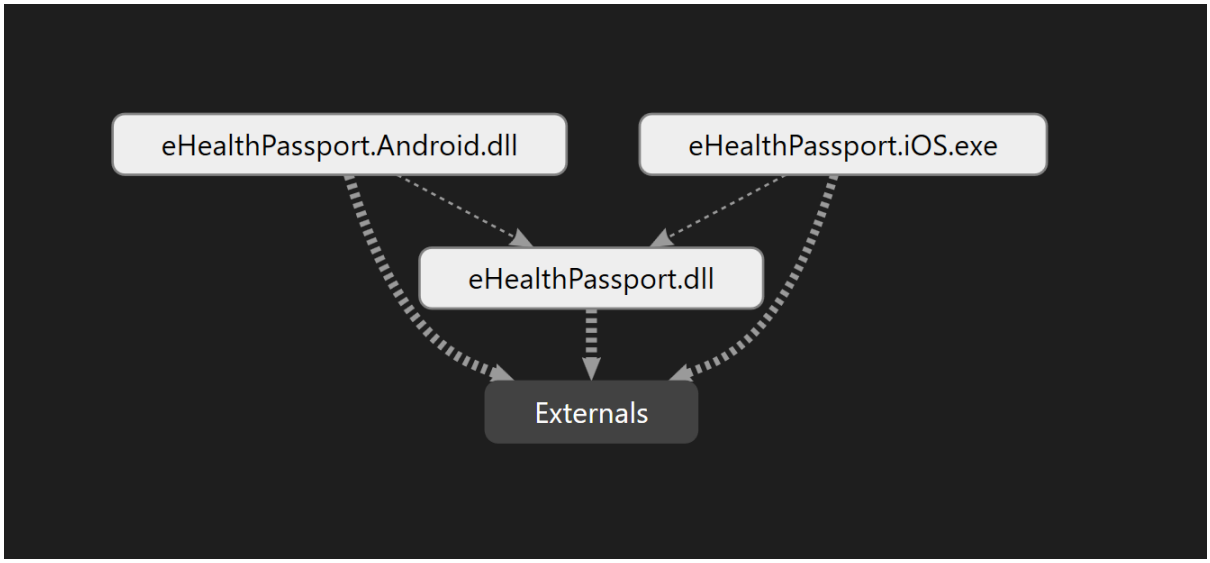


Figure 6: UML Class diagram

Figure 6 shows how each of the projects integrate with each other to create a native mobile application. It is seen, there is one central project that contains the vast majority of the codebase, whereas, platform specific code is held on each of the native projects; eHealthPassport.Android or eHealthPassport.iOS. To integrate third party libraries to build the project, C# requires NuGet to be used. NuGet is a package manager, which handles dependencies, updates, and installation of third-party libraries. The third-party libraries are represented in Figure 6 by the representation of Externals. It is seen, all three projects are required to have the NuGet packages added to ensure the correct dependencies are linked

Database Design

There are many choices to consider when designing how the app will integrate with the remote database. Two databases will be required; a remote database for the storing of all client data and a local database for storing of the single user's data. To integrate with the remote database, either database calls will be created on the app or a Web Service will be created to gather/insert data into the database.

A web service provides many bonuses to creating database calls in the app. If an app had 1 million users, this would mean 1 million separate apps will have access to the database consequently providing a maximum of 1 million active concurrent connections to the database. This could also mean there is the potential for up to 1 million connections consistently opening and closing putting a strain on the server.

Encapsulation of a database allows for developers and users to access the database without ever seeing or knowing what database has been implemented (Agiledata.org, n.d.). This allows for the database to be created, updated, replaced, and backed-up without affecting end-user devices as they will not require an update. Due to the encapsulation layer handling the database connection, the app will not require updating as it will still be pointing to the same URL or web service. Encapsulation layers can be implemented in a range of ways; however, it has become industry standard to implement them using a Web Service (Agiledata.org, n.d.).

Utilising a web service, provides an architecture that allows a developer to reuse both current instances and legacy systems by the use of a system integration. They provide a common place to execute all data-oriented processing that is to take place on the system. Whether this be: storing, inserting, updating, or deleting. Fault tolerance and error handling are catered for far better with the use of an encapsulation layer. Many common errors can be handled using an encapsulation layer (Agiledata.org, n.d.). For example, if the database is unavailable due to connection errors, it is possible for the encapsulation layer to cache the data until the database becomes available again for storage.

In view of these considerations, it has been decided to undertake the approach of creating a RESTful API to handle the database connections. This has been decided with the use of the above investigation and with the setup already in place at Liverpool John Moores University (LJMU) requiring access to the database to be undertaken on the LJMU network, behind its respective Firewall.

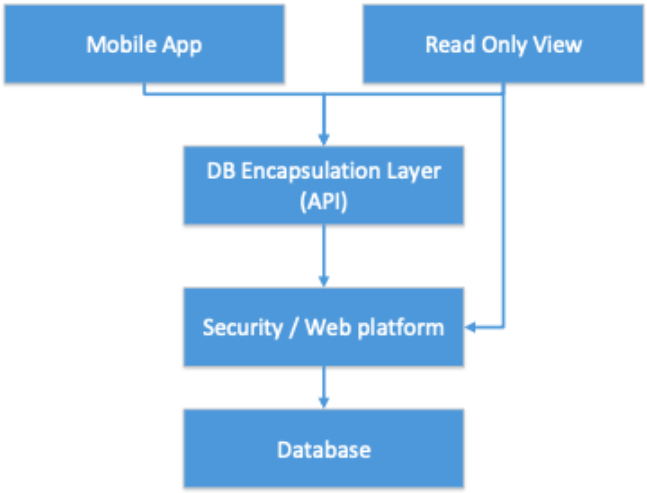


Figure 7: Diagram representing the proposed workflow of the app and read only view integrating with the database.

Figure 7 shows the proposed structure to integrate the mobile application with the app server. The DB encapsulation layer (API) will be created to accept data from the mobile applications. This API layer can then integrate into the existing security layer to ensure the data is secure and sanitised before persisting to the database layer on the App Server.

When designing a relational database model, it is expected to follow a normalisation process to ensure the database model is optimised. This is done by ensuring all data is dependent on the key that defines the table.

The database schema was moved through the normalisation process where it was then deemed normalised once reaching third-normal form.

First Normal Form

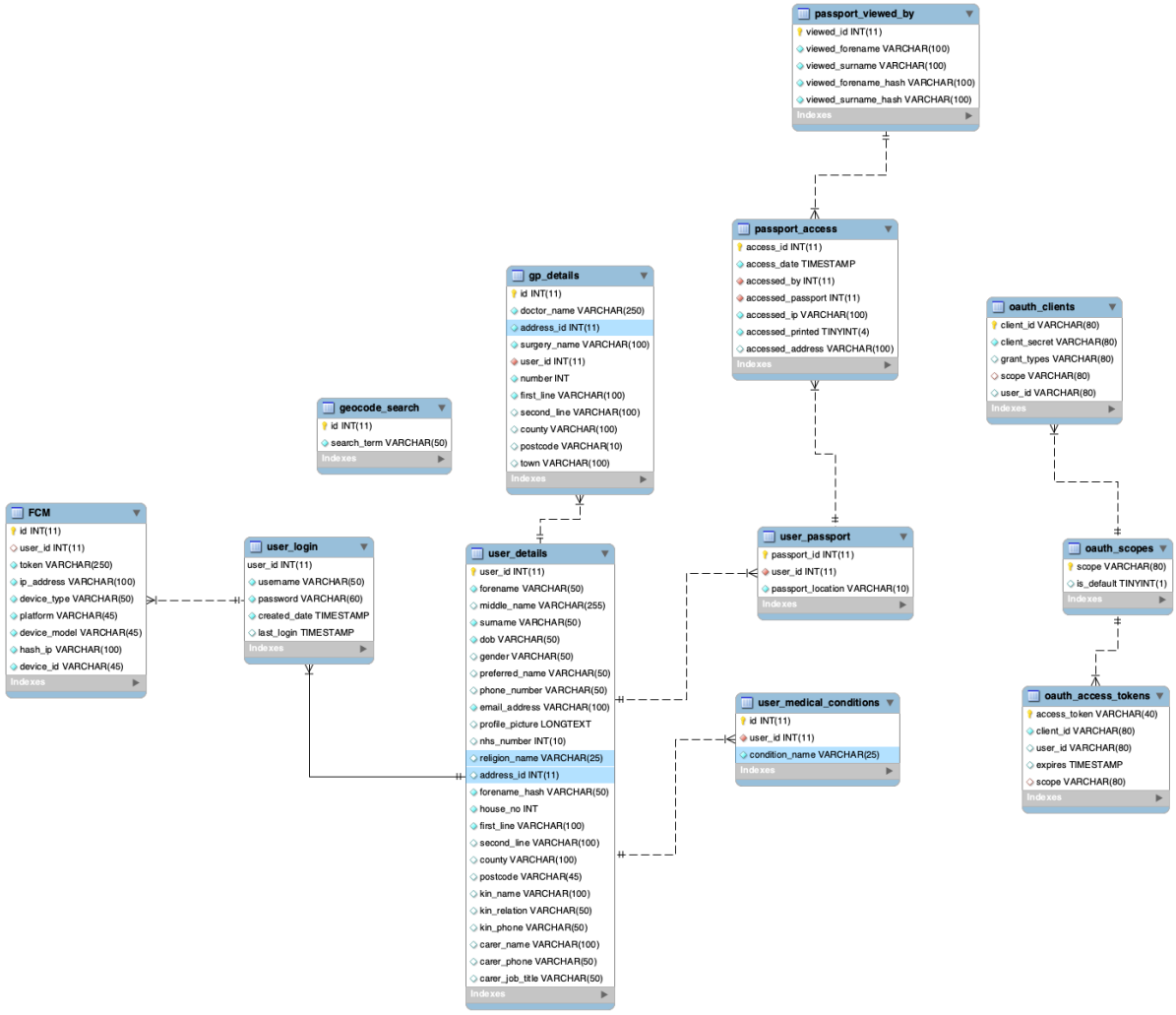


Figure 8: First normal form representation of the Database Schema for the App Server

Figure 8 shows the original schema proposed to create the database. As seen, there is one large central table that contains a large amount of redundant data. Tables like user_medical_conditions also see redundant data as this will create a large amount of duplicate data where users have multiple medical conditions or many users have the same medical condition, repetitive data will be seen in the table. These data redundancies hope to be fixed with the next step in the normalization process; second normal form.

Second Normal Form

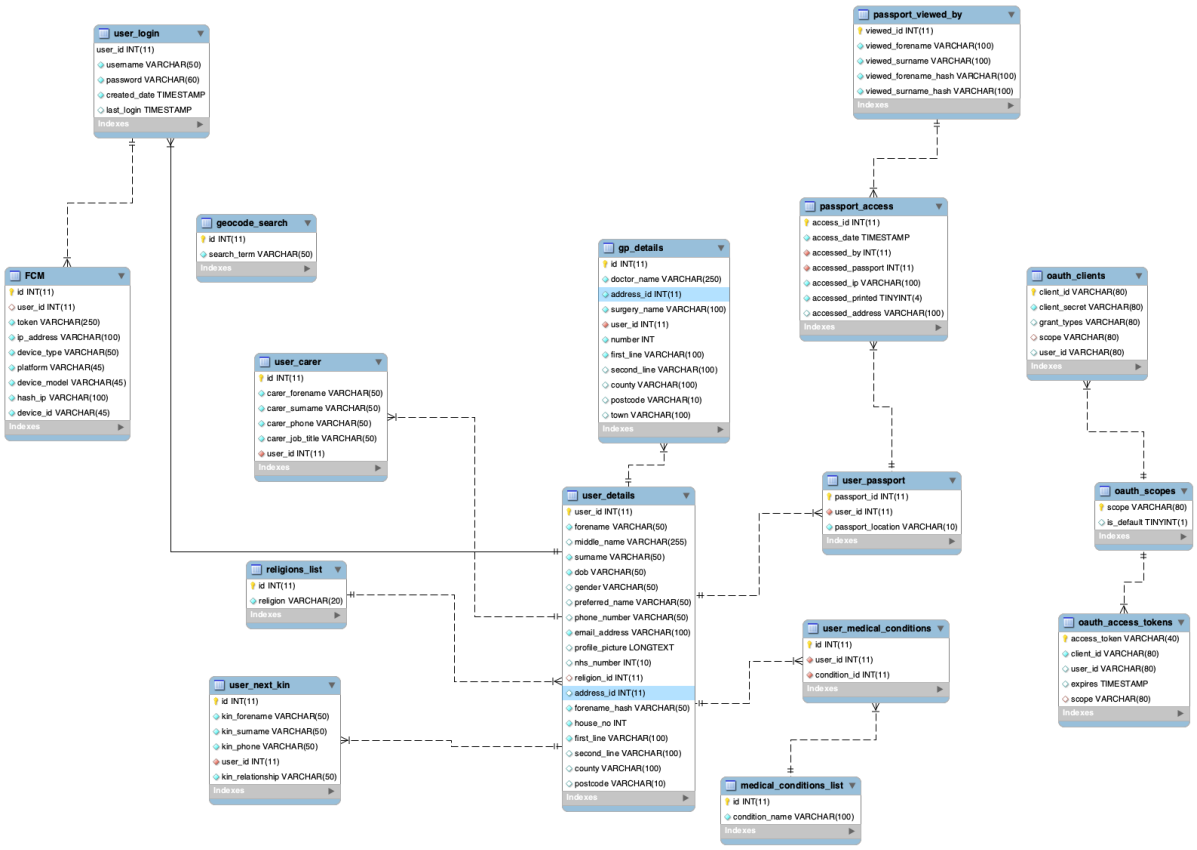


Figure 9: Second Normal Form representation of the database schema to be created on the App Server

Second normal form sees the removal of these data redundancies that were discovered in first normal form, the new database structure can be seen in Figure 9. Next of Kin and Carer details have been moved to separate tables with religions also being moved. Secondly, a list of medical conditions has been created that sees a foreign key to the user_medical_conditions. This means that memory and data redundancies can be saved by using an integer value in user_medical_conditions rather than a string value.

Creating the next of kin and carer tables also now means, in theory, it is possible to allow a user to have more than one carer or next of kin specified in the ehealth passport.

Third Normal Form

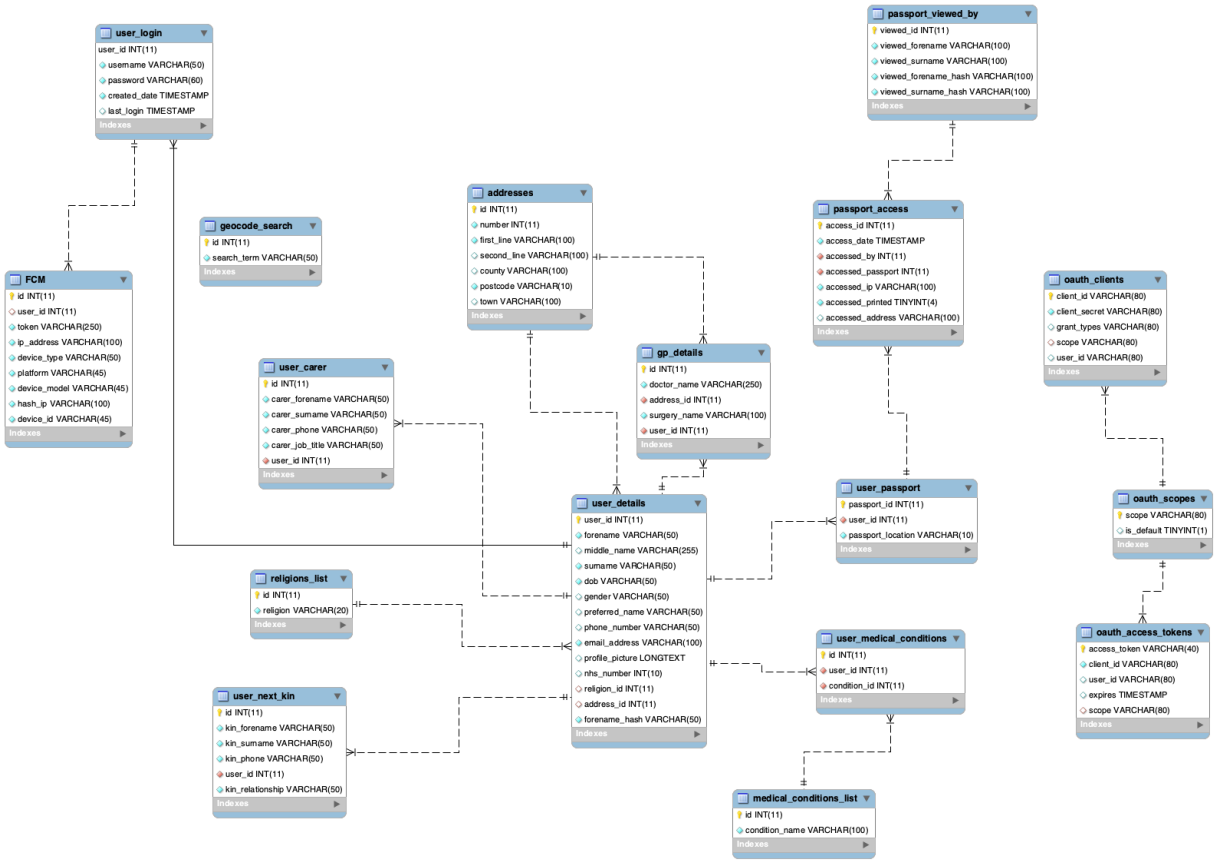


Figure 10: Entity Relationship Diagram for global database to be held on the Main Server.

Figure 10 demonstrates the final step, third normal form. Third normal form ensures all data, in each table, is dependent on its primary key. To reach third normal form on this database schema, postal address data was moved out of the user_details and gp_details tables into its own table. This will inevitably save data were addresses can relate to many different users.

The database structure, which is in third-normal form, is represented in Figure 10. This ERD diagram shows the relationships between each of the tables and the data structure for the global database. An embedded SQLite database will be created to mirror this structure although some of the table names may be changed to match JSON or a C# naming convention. It is also unnecessary to adhere to the foreign key constraints on the embedded SQLite database due to only one user's data being stored on the app.

Location Services Design (Geocoding)

Geocoding is the process of converting a plaintext address into Longitudinal and Latitudinal Coordinates (Google Developers, n.d.). With such a vast amount of geocoding API's available, programmable web has more than 800 mapping API's available for use, however, it is hard to know which the best geocoding API is. Each different API provides different advantages and disadvantages, including some being open source and others pay to access.

Google Maps

Although the main Google Maps API has been disbanded, multiple API's have been created from it. The geocoding API provides geocoding and reverse geocoding solutions to convert addresses. The google maps API can easily be accessed from Apps and Web Applications via a HTTP interface. Using a RESTful API, it is possible to send a GET request to the Google Geocoding API to Geocode and Reverse Geocode an address. Using the reverse geocoding API, Longitude and Latitude coordinates into a human readable address. Different types of fields are returned to the requester so the response can be processed. The Google API provides fields to allow for the Street Address, Type of Building, Postal Code, and Location type to be processed. By processing the 'Type of Building' field, it can be determined whether the device is located within or near a healthcare building. This is not an Open Source or Free to Access API. Google's API is provided at a 'pay for what you use' structure. This provides varying costs where you pay for the amount of processing power used.

Using a GET message, it is possible to convert longitude and latitude coordinates into an address; consisting of a name and a type. The type value can be used to establish what the type of the building is. The address can be used to determine what the name of the building is. However, although this API provides a very in-depth overview of an address, it is not open source. Google provides access to their API's with a 'pay what you use' structure. This structure provides access to the API for varying costs according to how many calls are received (successful and unsuccessful calls).

SKU	$200 MONTHLY CREDIT EQUIVALENT FREE USAGE	MONTHLY VOLUME RANGE (PRICE PER THOUSAND)		
Other types of requests related to places				
Geocoding	Up to 40,000 calls	$5.00	$4.00	
Geolocation	Up to 40,000 calls	$5.00	$4.00	CONTACT SALES
Time Zone	Up to 40,000 calls	$5.00	$4.00	for volume discounts.
Elevation	Up to 40,000 calls	$5.00	$4.00	

Figure 11: Pricing structure for the Google Maps Geocoding API

As seen in Figure 11, the pricing structure varies depending on the number of requests made (not that are planned). Google does provide a free tier for the first year of holding an account. This comes with $200 of free credit. As this project does not require real users and the Geocoding API will only be used for testing, it is projected the $200 of credit will not run out.

An example of the response from the Google Maps API can be seen in Figure 12 As seen, Latitude and Longitude coordinates are converted into an address. This address can be processed to determine whether the user is within or near a healthcare facility, like a hospital. It is possible to either process the long_name, short_name and/or the types field when processing the address. It will be required to process these to see whether they contain keywords/phrases, which will be provided beforehand.

Figure 12: Example of a response from the Google Maps Reverse Geocoding API.

OpenStreetMaps Nominatim

OpenStreetMaps (OSM) is an open source mapping API. A geocoding API is provided called Nominatim, which is also open source. This API once again converts longitude and latitude coordinates into an address. However, it provides many more details. Nominatim provides a key, when existing, it can be seen the location is a hospital. A key is also returned, which determines the type of the building at the location. This value contains a string to determine what the building is. This value, for example, could equal ‘hospital’. However, unlike the Google Maps API, it only returns the first result the API can find. This, therefore, is not as accurate as the Google Maps API which returns a List of Locations. However, OSM Nominatim has an increased accuracy in their type value which could be leveraged. As seen in Figure 13, only one type is returned from the API, however, it is more often correct when compared to the Google Maps API. Although, due to Nominatim being an open source project, there are limits to the number of requests, which can be made to the API. This limit is not made available on their web page however, they do claim the limits are in place.

```
{
  "place_id": "91896945",
  "licence": "Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright",
  "osm_type": "way",
  "osm_id": "68099283",
  "lat": "53.40946035",
  "lon": "-2.96459755508448",
  "place_rank": "30",
  "category": "amenity",
  "type": "hospital",
  "importance": "0.337229093694288",
  "addresstype": "amenity",
  "name": "Royal Liverpool University Hospital",
  "display_name": "Royal Liverpool University Hospital, Prescot Street, Islington,
Liverpool, North West England, England, L7 8XP, UK",
  "address": {
    "hospital": "Royal Liverpool University Hospital",
    "road": "Prescot Street",
    "suburb": "Islington",
    "city": "Liverpool",
    "state_district": "North West England",
    "state": "England",
    "postcode": "L7 8XP",
    "country": "UK",
    "country_code": "gb"
  },
  "boundingbox": [
    "53.4077251",
    "53.4107138",
    "-2.9679176",
    "-2.9613934"
  ]
}
```


Figure 13: Example response from the OSM Nominatim API where a reverse geocode has been done on the Royal Liverpool Hospital.

Xamarin Essentials Library

Xamarin Essentials Library is a Cross Platform API, which provides many interfaces for use with iOS, Android and Universal Windows Platform Apps (Docs.microsoft.com, n.d.). The Xamarin essentials library provides access to a Geolocator, a Placemarks API and the ability to calculate the distance between two locations. For this project, the placemarks API could be used to convert Latitude and Longitude coordinates into an address, which could be processed. It is possible to use the Xamarin Geolocator API to get the last known address of the device. Due to this solution being a native API, this means no HTTPS calls are required meaning this could process faster than an API, which would require a HTTPS call (as it will be dependent on network connectivity and speed). However, less information is returned for processing of the address. Unlike previous Geocoding providers, only the address is returned (Example: Microsoft Building 25 Redmond WA USA (Docs.microsoft.com, n.d.)) there are no tags provided, which could be checked. This would require the first-line of the address to be checked for keywords, which will be provided by a developer. This, however, is not an effective solution. This requires all healthcare facilities to provide what the facility is, in the first line of address. Many hospitals only return their address, which does not contain the key words that would be searched for. For example, Alder Hey Hospital would return the following string value: "*E Prescot Rd, Liverpool L14 5AB"*. To process this, it would require a source of supported healthcare centres to be provided to process the location, although this could be done, it is not an adequate solution for this project due to its timescale.

Location Services Activity Diagram

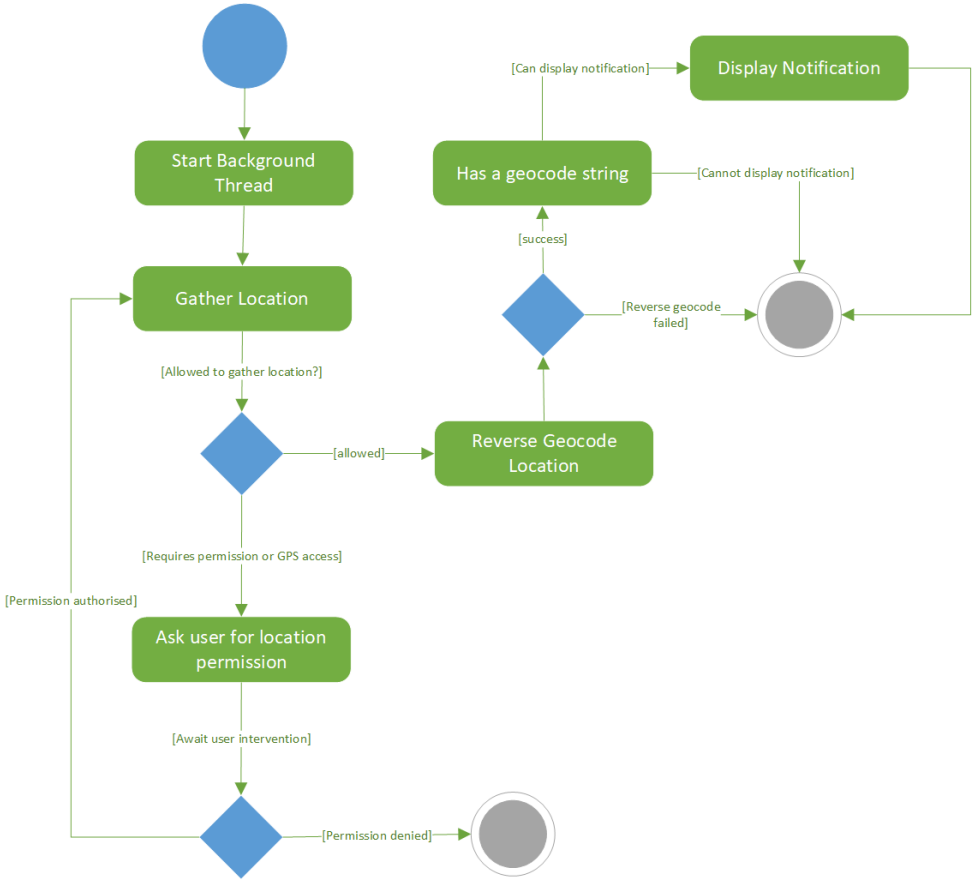


Figure 14: Activity diagram to demonstrate how background tasking is used to gather the location of a device.

Figure 14 demonstrates how a background task is to be used to derive the devices location, perform a reverse geocode, and if possible, display a notification. Firstly, the background task is started and once successfully started, the device will start to gather its location. This, however, could require a decision from the user. During the first instance of this background thread running, the user will be prompted to give the device permission to access location services. If the user chooses to give the application access, then this will remain until the user revokes the permission. Once the permission has been granted, location can be gathered from the device. Then, it is possible to reverse geocode the location, with one of the API's discussed above.

If one of the key words is found to be within the reverse geocode, then the notification will be displayed on the device. Otherwise, the process will be ended.

Mobile Wireframes

Figure 15: Wireframes displaying the proposed designs for the Cross Platform Mobile Application

As seen in Figure 15, all possible mobile pages have a wireframe template drawn, which will be created when the mobile application is implemented. Navigation is represented with the use of arrows. Each arrow originates with the UI element, which will trigger the navigation. Some navigation arrows, however, are not triggered by a UI element and these are represented by an arrow originating from the background. For example, the lock screen template appears from the user navigating to the App so this template will not require a UI element or user gesture to trigger.

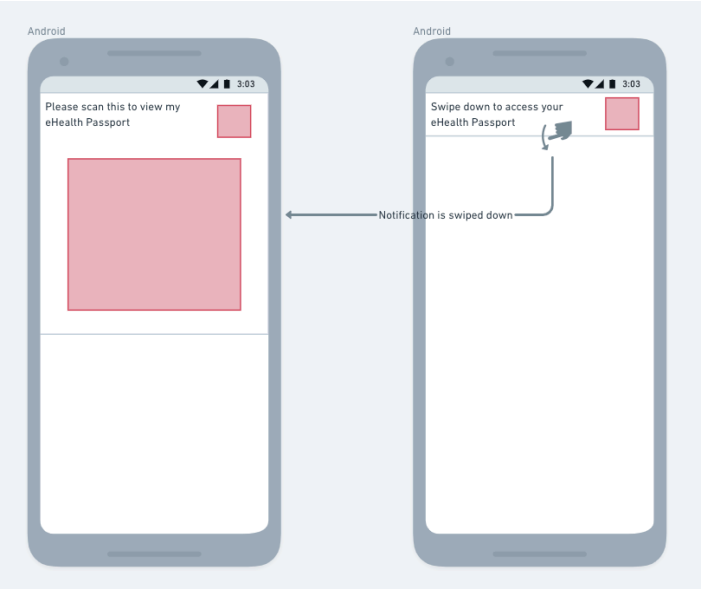


Figure 16: Notification layout design.

The wireframe in figure 16 lays out a notification, which will display a QR code to a user whilst their device is locked/unlocked. The difference between locked and unlocked will be the position of the notification. When the device is unlocked it will be displayed at the top of the screen and when locked, it will be displayed in the centre. A swipe action is to be added to the notification. If the user swipes down on the notification, then the QR will become a large size so that it is easily scanned by a user. This can be seen in Figure 16 by the representation of the red square.

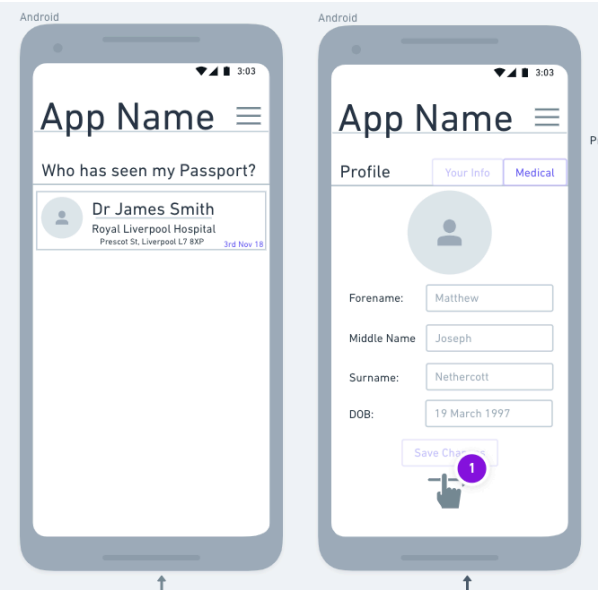


Figure 17: Two templates are designed to hold data on the mobile application

The wireframe designs see the design of two main templates: a List and a Form design. Consistent designs and templates across the app will ease in the implementation process and provide a similar user experience across the app for the user.

Each item in the list will be wrapped in a custom view where important information will be displayed to the user. This data includes: an image, title, datetime, and a description. A List will typically be used when the user is not required to enter any data but only view the data.

Forms will be used when the user is required to provide data and view the data. An example of this would be on their profile page, where the user will be required to update their data. Typically, no more than six inputs should be provided on each page. To minimize the number of inputs, a carousel page design will be used to allow a user to swipe across each of the pages to discover further inputs. Tabs will also be used to separate information such as clerical and medical information, which is seen on one page.

Web Wireframes (App Server)

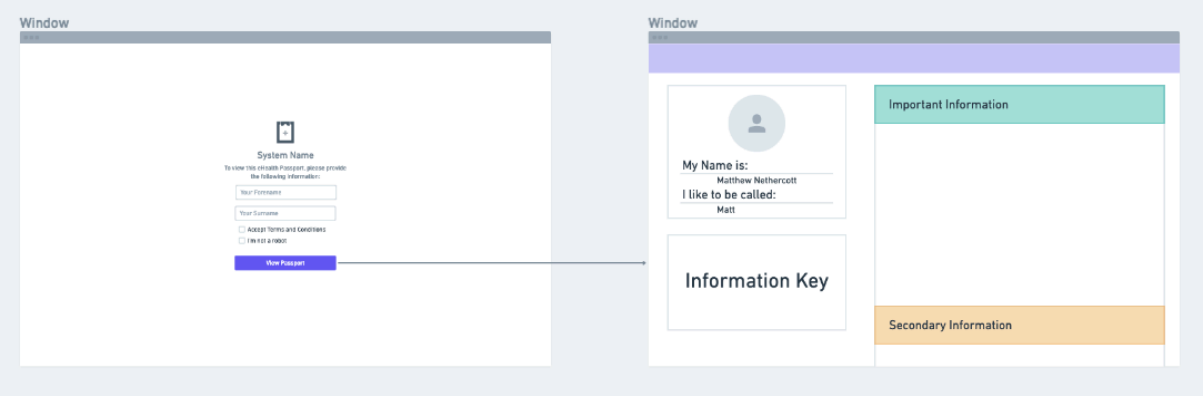


Figure 18: Wireframe to show the proposed design of the Read-Only view of the user's eHealth Passport

Figure 18 displays the proposed layout of information a healthcare professional or someone viewing a health passport will see. Information is to be split into three main areas, Base information, a key, and then the main information.

Base information is defined as key information on this user: Name, Photo and Preferred Name.

The key will be used to help inform a user on the information they are seeing. It is also used to inform them on how well the information has been filled out and, any consent given.

Lastly, the main bulk of the information will be displayed on the right-hand side of the page. Only information that has been provided by the user is to be displayed on this pane, with the most important information being displayed at the top of the page.

The page will be designed, so information displayed in the key and the base section will scroll with the user, whilst they read the information in the main sections of the layout.

Security Design

Although, the creation of this project will only handle test data during its creation. It is planned for it to eventually handle sensitive medical data. With this expectation, it is required to ensure the data, during transmission and rest, will be secure. Although, HTTPS provides end to end encryption of data during transmission, it will be required for an additional layer of security to be applied on top of this. The choice to use the LJMU Server (CMPPROJ) is the reason for this. CMPPROJ still uses the previously deprecated TLS 1.0 (currently superseded by TLS 1.3).

The following security measures will be taken to ensure that the data is secure:

- All data connections to outside sources must be served over HTTPS
- All data must be encrypted whilst stored on the Mobile Application's embedded database. Jailbroken or Rooted devices will have access to the Embedded Database held on the phone so data stored must be secure in the event of a user losing their device.
- Any Personal Identifiable Information must be encrypted on the main database held on the App Server. Developers will have access to the database, consequently meaning that they could potentially have access to sensitive medical data.
- It is required that each time the user navigates away from the Mobile Application, that the Mobile Application 'locks' itself and requires the user to provide their password again to unlock it.
- Where sensitive data is to be persisted on the Mobile Application (stored for session data) it must be stored using a secure storage such as Keychain.

Encryption Design

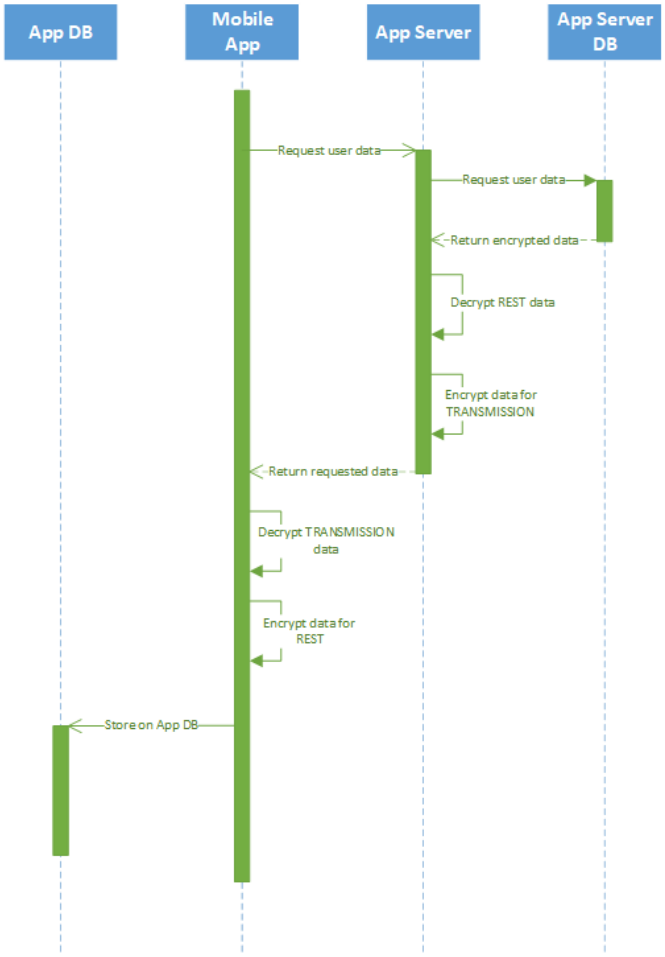


Figure 19: Sequence diagram showing the workflow of data transmission between the App Server's database and the Mobile Application's database.

As discussed previously, encryption is vital for healthcare applications to ensure personal identifiable information is kept secure and protected on the application in the case of theft, unauthorised access, or unauthorised manipulation. An AES encryption algorithm has been chosen due to its rigidity and strength. A random initialisation vector (IV) is to be generated and applied to each encrypted string. Randomisation is required for an IV to allow for an encryption to reach semantic encryption. A random IV allows for each string to be encrypted differently. The encryption of the String “Liverpool” will encrypt differently each time it is encrypted due to the random nature of the IV given. Due to its nature of being random, it does not need to be kept secret and will be required by the system decrypting the string to decrypt the string. This means the IV will have to be sent, along with the encrypted string. In this scenario, the IV will be appended to the encrypted string with a certain character used to split the two.

As seen in Figure 19 three different encryption keys are to be used; two encryption keys for storing data at rest (one for App Server DB and one for App DB) and then one last encryption key for data during transmission. This setup, however, will require the transmission encryption key to be held on both the mobile application and the app server. This is not an ideal setup, if the App was to be reverse engineered a malicious user could gain access to the encryption key. However, this will be accounted for; with ProGuard, and signing the app with a certification on iOS. ProGuard is a tool for Android, which allows for the obfuscation and linking of Java code (Microsoft, 2019). This makes it very difficult for a malicious user to gain access to the encryption keys. Further steps can also be taken to hide the keys during runtime of the mobile application. On Android, these steps include:

- Using ProGuard,
- Obfuscated/Encrypted Strings,
- Hidden using Native Libraries

With all these steps adhered to, it is no longer possible for a malicious user to find the location of the encryption key in the memory of the application.

Secure Storage

Secure storage will be used where sensitive data needs to be accessed quickly. This allows for the storage of persistent data on both iOS and Android. Secure Storage is encrypted in Keychain for iOS and encrypted data is stored in Shared Preferences with keys in Android KeyStore. Any sensitive data about the user will need to be stored in Secure Storage over Application Properties. Application properties persists data, however, it is not encrypted and stored in plaintext. This makes it ideal for non-sensitive data, where processing power should not be used to encrypt strings. Table 3 displays, which data is to be persisted in Secure Storage and which data is to be in Application Properties.

Table 3: Data to be persisted in Secure Storage and data to be persisted in Application Properties

Secure Storage	Application Properties
Access Token	User ID
Username	

User ID has been deemed unnecessary to store in Secure Storage as, to use the User ID value to access any data it has to be paired with a valid, active Access Token. Consequently, it is best to keep these separate in case one of the stores is compromised. As seen, minimal data is to be persisted in these stores, as the embedded SQLite database is to be used for the majority of data.

User Stories

The use of user stories is vital in Agile development to allow developers to refine requirements into a set of steps, which can be implemented easily. They are often refined by the type of user that someone can be in a system. This project will see two different types of user; patient and healthcare professional. User stories will help to guide clinical information for developers of this project. Often, developers do not understand requirements passed to them. Especially if they are not within their field of expertise. This section will see the definition of user stories to help implement the guide the implementation of the project.

User stories will be outlined in the following format: As a [user], I want to be able to [do something], for [reason]. This allows for easy development of requirements as they have been refined into stories which can be developed with an agile methodology.

Patient

As a patient, I want to be able to record my next of kin as they are often able to provide further details on my conditions and treatment.

As a patient, I want to provide my carer information as they will be able to also provide further information about my conditions and treatment.

As a patient, I want to be able to provide information regarding my personal care so that I can be cared for in the way that is considerate to me.

As a patient, I want to be able to provide information on how I communicate so that professionals can communicate with me in a way that is effective.

As a patient, I want to give information on how well I can see in hear to make professionals aware.

As a patient, I want to be able to provide information on food and drink that I like so that food is catered to any dietary needs.

As a patient, I want to be able to give information on how I take medication and go to the bathroom so I can get the correct level of support.

As a patient, I want to report any details about my movement ability so nothing is required of me that I am unable to do.

As a patient, I want to be able to provide details on how I manage pain so that my treatment can be tailored to my ability.

As a patient, I want to be able to provide my sleeping routine so that I am not disturbed during my sleep.

As a patient, I would like to report any information that could make my stay better or worst so that health care professionals can tailor their experience to me.

As a patient, I would like to give information on how I seek comfort so that this can be provided whilst in hospital to make my stay comforting.

As a patient, I would like to report any details on support that I require so that health care professionals know.

Health care professional

As a health care professional, I want to be able to see a patient's name, NHS Number, address, and DOB so that I can identify them within our existing health system.

As a health care professional, I want to see a patient's phone number so that I can contact them when they are not in for any updates that they may require.

As a health care professional, I wish to see the most recent medical history so that I know what recent treatment a patient has had to effectively treat them now.

As a health care professional, I want to see what medication a patient is currently on so that I can prescribe medication that will not affect medications that the patient is currently on.

As a health care professional, I want to see what GP surgery a patient belongs to so that I can see their medical information.

As a health care professional, I want to see what allergies a patient has so that I do not prescribe any medications that may clash with them.

As a health care professional, I want to see any risks of safety issues that revolve around the patient so that they are safe and secure within my treatment.

Implementation and Testing

Background Task Implementation

Since the design phase, it has been discovered, background tasks have been limited on the Android platform since API 26 (Android version 8.0) (Android Developers, n.d.) however, it is still available on the iOS platform. This, however, provided an issue with development due to the reliance on background execution to determine location. A workaround was discovered with the use of Firebase Cloud Messaging.

Firebase Cloud Messaging allows for a background task to be run if the device receives a high-priority data message from its service (Android Developers, n.d.). Sending a high priority data message from FCM, provides the app with temporary access to a background task. Using this, it is possible to derive the device's location when it receives this message and display a notification, if necessary. However, it is still needed to run periodically on a timer to ensure the device's location is checked regularly. On an Apache server, it is possible to run a CRON job periodically to send a data-message to the devices by using the Firebase Cloud Messaging Service.

Without access to create a CRON job on CMPPROJ (or a scheduled task), a CRON job was setup on a separate server to send a GET request to an API call created on the CMPPROJ server. This API call sends a data-message to the Firebase Cloud Messaging service to send a high-priority message to every device that is registered and has a user logged into the mobile application. To meet the requirements defined, this CRON job was run every 15 minutes with the hopes of this extra processing not affecting the battery life of the mobile device. During the testing phase, it was discovered the extra processing did not introduce any extra strain onto the device. Less than zero percent battery life was used (by the app) during the period of seven days.

```
{
  "topic": $deviceID,
  "registration_ids": $tokens,
  "priority": "high",
  "data": {
    "passport_location": $location,
    "emergency": true|false,
    "search_terms": null|array[String]
  }
}
```

Figure 20: Typical payload sent to Firebase Cloud Messaging

Figure 20 displays a typical JSON payload, which is sent to start the geocode/notification process on a device. Table 4 explains and describes all necessary variables, which are used in the payload so that FCM can correctly process the data. The data object is passed directly to the device. This is where the critical data is required to process the geocode and notifications will be placed.

Table 4: Description of required data to correctly implement the FCM with the Xamarin Mobile Application

Key	Description
Topic	What notification stream do we wish to target? It is possible to subscribe multiple devices to the same channel to target them all.
Registration_ids	Unique identifier for if we want to target certain devices. In this case, only one device will be provided due to individual passports.
Priority	The priority of the data message. High priority will be used to force the device to process the data using a background thread.
Passport_location	The identifier for the passport location to add to the URI to identify the user.
Emergency	If the emergency access passport was used this will allow the device to skip the geolocation checks.
Search_terms	String array of key terms to use when string matching the geolocation payload.

Workflow designed and implemented

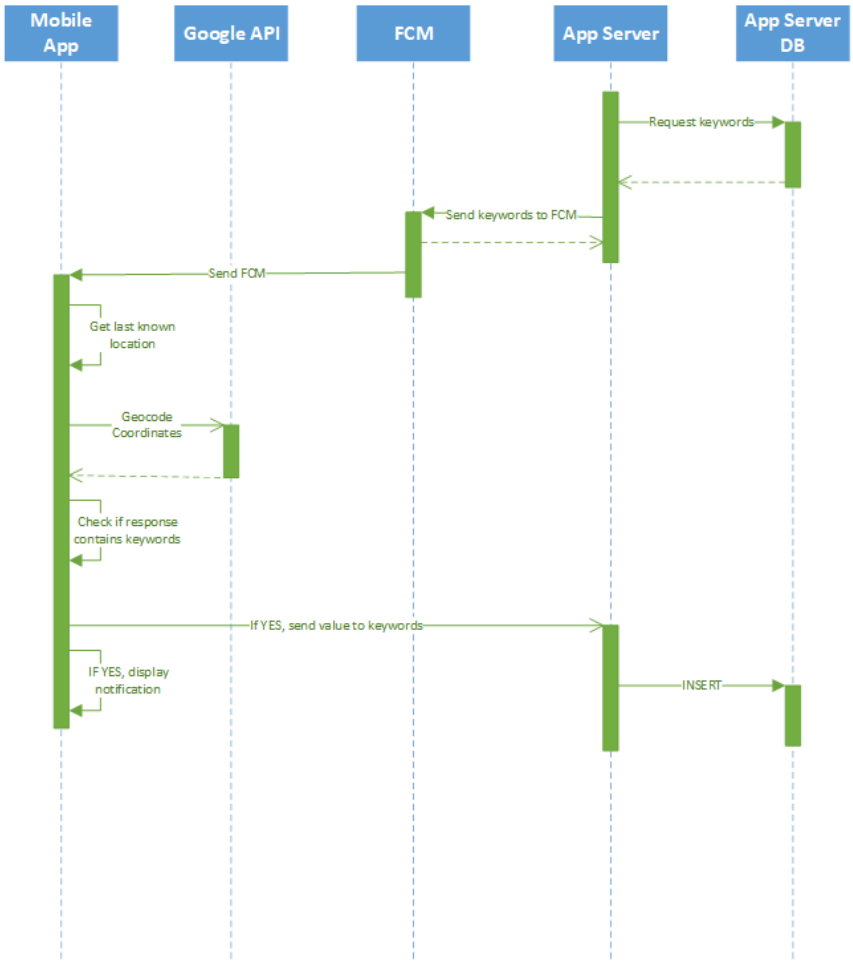


Figure 21: Workflow designed to implement Firebase Cloud Messaging, derive Location, display a notification, and refine the algorithm.

Figure 21 outlines the workflow followed by the app server, FCM, and mobile application to: derive location, perform reverse geocode, and display a QR notification. As seen, the workflow starts on the app server by loading the key words from the database (or knowledge base). These keywords are then loaded into a structure, which can be sent via FCM to a mobile device; the JSON payload in Figure 21. This payload is then forwarded to the mobile device being targeted alongside their passport information and whether the request is an emergency (in this case this would be false). The last known GPS location of the device is then fetched and reverse geocoded using the Google Maps API. Originally, the Xamarin Essentials API was used, however, this was seen as inaccurate in some hospital locations where only a street address was found. The response is then checked to see if it matches any of the keywords in the knowledge base that was sent. If a partial match was found, this can be used to update the knowledge base on the App Server for an increase in accuracy in the future. Allowing the algorithm to refine itself over time. This would be sent asynchronously allowing for the notification to appear on the device without any further delay.

Mobile Application

Login and Locked

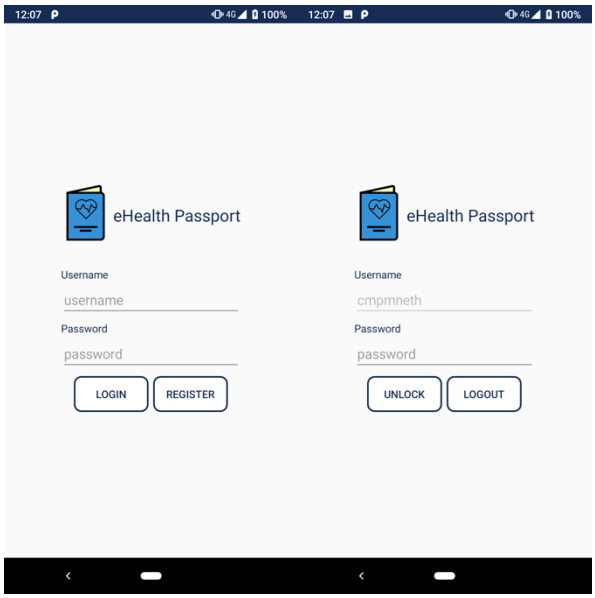


Figure 22: Unlock and Login screen implementation. Uses the strict guidelines set in the initial wireframes.

When a user first uses the app, or uses after logging out, they will be required to login to the mobile application. The login screen (represented in Figure 22) is also displayed once the App has 'locked' itself, which happens every time the focus is taken off of the Mobile Application. This means, whenever the app is minimised, moved to the system tray, or the mobile device is locked the created lock screen will intercept the user when bringing focus back to the app. Once the user has pressed either the Login or Unlock button, the mobile application will encrypt the username and password to be sent to the App server. The App server then checks the stored hashed value in the database. If correct, a client credential grant will be issued to the created oAuth 2.0 server. oAuth 2.0 has been used to prevent client details consistently being sent with every API call to the App Server. This increases the application as, if an API call is intercepted, it is unlikely to contain client credentials as these are only available when logging in/unlocking the device.

The access token returned to the mobile device is designed to last 1 hour from creating. This means, in the unlikely event it is intercepted, a malicious user only has a maximum of one hour to work out how the API works to utilise the access token. Access tokens are also encrypted so the identity of it making it harder for a malicious user to know how the token works.

Once this process has been performed and the device receives an access token, the access token is persisted into Secure Storage (using the operating systems encryption). This is then required for every API call that returns or alters any user data.

QR code and Privacy

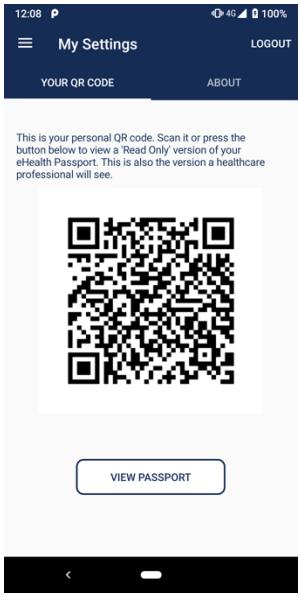


Figure 23: Page used to display to a user their QR code so that they can show it to a health professional.

Due to the nature of the project using QR codes for access, a page allowing a user to manually show their QR code has been created. This means, in the event the notification is not displaying, a user can display this page to a health professional, so they are still able to scan the QR code. To generate the QR code, a third-party library was used; ZXing. ZXing is a barcode generating and scanning library originally designed for Android but has been ported for use with Xamarin.Forms (Owen, n.d.). This allows a QR to be generated to open a URL. This URL is constructed using the URL for the app server, a unique identifier for the passport, and any security information required.

From this page, it is also possible for a user to click a button to open their read-only passport on their device. This link is created from the same URL that is used to generate the QR code and, will display the same read-only view.

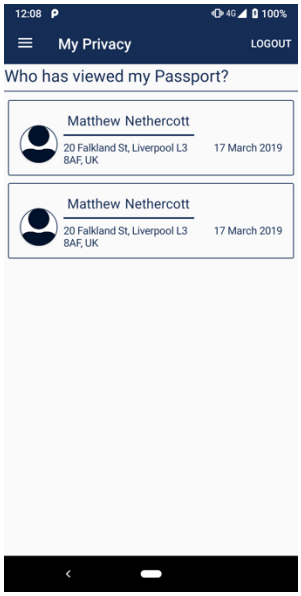


Figure 24: A privacy page was created to show a user who has viewed their read-only health passport

When someone views a health passport (their own or someone else's) they are required to provide their forename and surname. This is logged and sent to the user's device. This is then displayed on the user's device in the privacy section, which can be seen in Figure 24. This allows for the user to see who and where their data was accessed. As the user, viewing the passport, will be prompted to provide their location, if they choose not to, this will also be displayed to the user. The date is then displayed alongside the information so a user can see when their data was accessed. The list is then ordered with the most recent first. In its current state, the user is not able to see any further details, however, with further work into the mobile application, it is hoped the user will be able to see specifics on what data was scrolled to and viewed. This would be done by adding a modal that would appear when clicking on a specific entry.

Profile Page

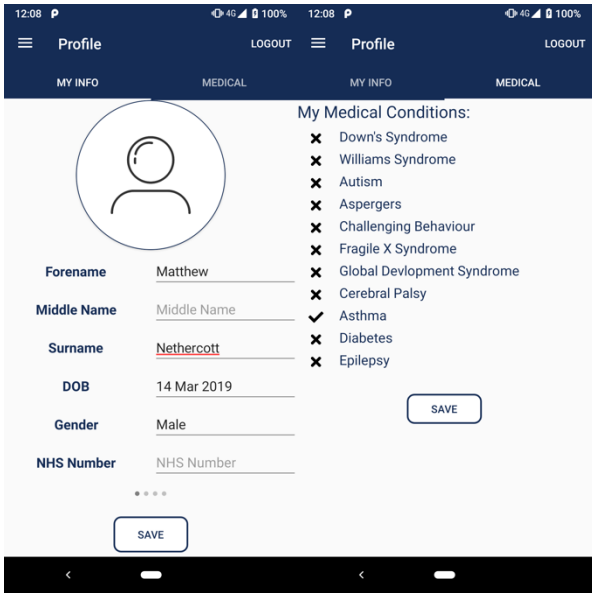


Figure 25: Tabbed layout with to display a user their personal information. This consists of basic profile data, next of kin, carer, and medical conditions.

The profile page is built using both a tabbed system with a carousel view. This allows for the amount of information on each screen to be minimised without, losing any clinical value. The user can quickly use a swipe gesture to navigate through the page. Both the Tabbed and Carousel views can be seen in Figure 25.

During the creation of the app, a profile page was added to provide basic information about the user. This includes, Base information, Next of Kin, Carer information, and Medical Conditions. Each base information page displays a number of text boxes and a save button on the page. There are never more than 7 text boxes to help prevent overwhelming any users; especially with the targeted audience. A swipe gesture is used to minimise data to a page to aid this. A save button is added to the bottom of the page; this button is only enabled once the user has updated any of the details. These details are saved to both the SQLite database and the App Server's database. Currently, there are 4 pages on the "My Info" tab of the profile page. The first two pages form basic information about the user. The third and fourth display further information, such as, Next of Kin and Carer.

A custom ListView was created to handle the medical conditions. This list is built from data fetched from the API. This list includes all medical conditions supported by the passport, with medical conditions the user has specified as having. This data is persisted and stored in the SQLite database and each time it is updated, it is updated in both the SQLite and on the App Server. There are two icon types used in this ListView – both of which are SVGs provided from the Font Awesome Library. Font Awesome is an open source vector library, which can be used on Web, and Mobile Applications (Fontawesome.com, 2019). The first of these icons is the tick icon, this indicates a user has selected this medical condition alerting they have it. The second, is the cross icon. This indicates the medical condition has not been selected. When a medical condition in the list is clicked, the icon next to the medical condition is updated to its new icon. It is only updated in both the databases once the

save button is clicked at the bottom of the page. With further development time, a search function would be added to refine the medications list. It is foreseen, with a growing number of medical conditions being added to the system, in its current version, this list would be overwhelming.

This list is currently database driven from the App Server. This means, as the list of supported medical conditions grows, the app will not need to update as it comes from the API. Currently, a list of common learning disabilities provided by Mencap has been used to drive the list. Mencap also defines that people can live with more than one learning disability, meaning the provision to choose more than one medical condition was implemented.

Read Only View

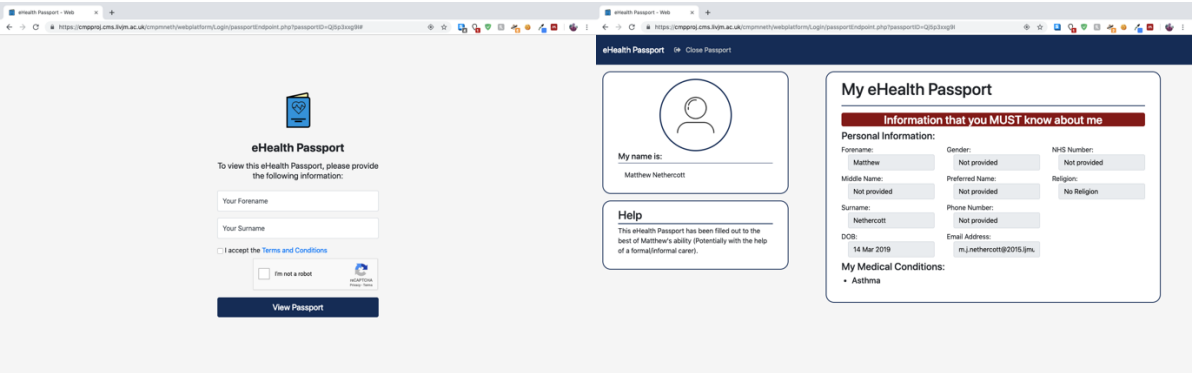


Figure 26: Read only portal held on the App Server. Figure demonstrates the workflow from providing details to, seeing user data. Also displays what details the person could possibly see and how.

A read only view was implemented to give a person read only access to a patient's health passport. This read only view can be seen represented in Figure 26 with user details seen on the right-hand side. It was required for basic information to be gathered about the person viewing the data. Forename, and Surname is to be provided by the person. Location and IP Address is then automatically gathered for the database. Once this data has been provided, and the captcha has been checked, the passport is displayed to the user.

Most information is conditionally displayed, this means if the user has provided it, it will be displayed. Otherwise, none of the information is displayed, for this section. This is to stop information overload and to keep minimal amounts of data displayed. The only information that is always present is personal information on a patient. This basic information should allow a healthcare professional to ascertain enough information to find useful, clinical information on the patient.

However, if a patient chooses not to provide some information, this also needs reporting to a healthcare professional. For example, if the patient chooses not to provide their Middle Name. In this case, the null value in the database will be replaced with a string containing the value "Not Provided".

A help section has been provided in the bottom leftmost corner of the read only view. This is in place to help a healthcare professional understand how useful the information provided to them is. A hardcoded string was provided to inform a

healthcare professional (or anyone looking at the passport) the information is self-reported by the patient, however, they may have had help to provide this information.

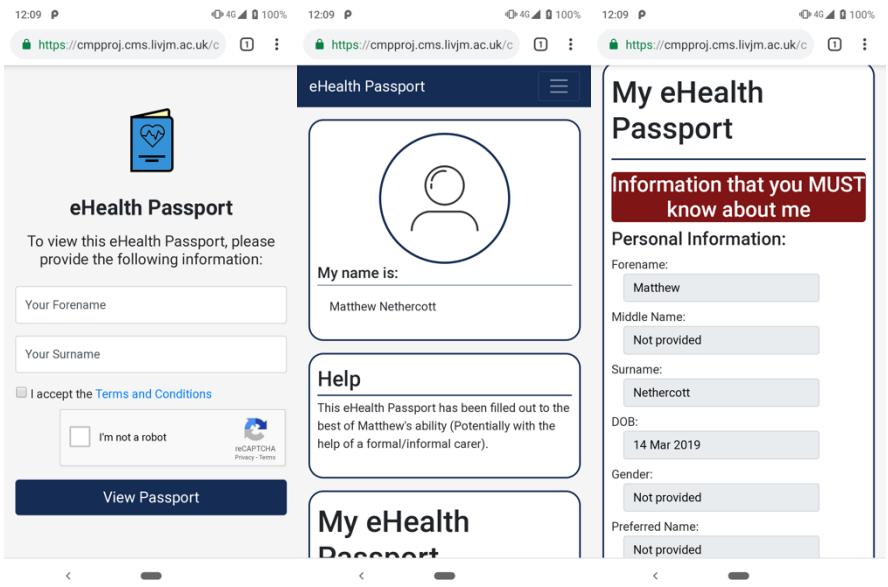


Figure 27: Responsive web design to allow a user to use their mobile or tablet device

Due to the nature of the read only view, being opened by scanning a QR code, it was required for it to be made responsive and to work on both tablet and mobile devices. As seen in Figure 27, the read only view is able to adapt to the smaller viewport of a mobile device. This responsive web application works on both types of devices where using responsive CSS and JavaScript, the web page is able to dynamically resize depending on screen size and orientation. All size calculations within the CSS are using the viewports width to calculate only minimal amounts of strictly set pixels.

The read-only view, just like the desktop web application, experiences the same behaviour. Elements like the help column, and traffic light markers are set to scroll with the user. As, especially with a portrait orientation, mobile devices will see a large amount of scrolling, it is important for the viewer to not be forced to constantly scroll up and down the application looking for data. By having this stick, scrolling with the user, means the user can see important information at a glance (requiring no extra scrolling). In the future, however, this could be improved. In the future, detecting the device type (Computer, Mobile, Tablet) could be ascertained to render certain elements on the page. A cookie banner would be used to display minimal information (Name, Link to consent, and Which section the user is on). This would allow for all important information to follow the user whilst scrolling, without the need for any information to be dismissed, like the current system.

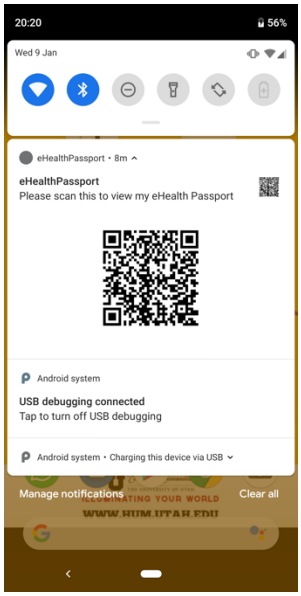


Figure 28: Image of the notification that is created and displayed when a device is detected to be within a health care facility

As seen in Figure 28 push notifications were successfully implemented from the wireframe designs specified in the design phase of the project. Figure 28 displays the notification after a swipe gesture has been used to display the large QR image.

To display the larger QR code, the zXing generated QR code is converted into a bitmap. This bitmap is then sized with a width of 512px and a height of 256px. It was found this is the minimum size of an image. This meant the image would fit perfectly into the notification without having to resize or have part of the QR code cut off.

Figure 28 also displays the size of the miniature QR code displayed before the swipe down gesture. As seen in the top righthand corner of the notification, a miniature notification is displayed to indicate to the user there is an image within the notification. The text is then different to indicate to a user they should swipe down the notification so they can display their QR code.

Emergency Access

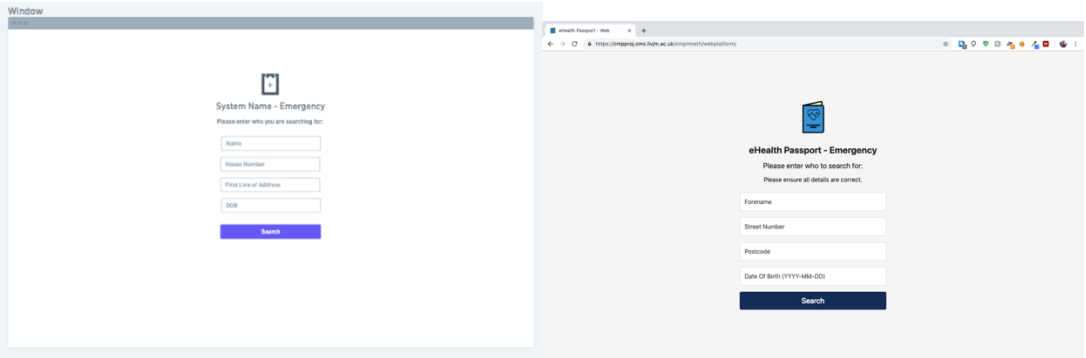


Figure 29: Search engine designed and created to provide emergency access to a QR code on a mobile device.

During testing of location services, it was realised, although location services for when a patient enters a healthcare facility solves many issues residing with Health Passports, it does not solve an issue surrounding point of contact. Emergency services are often called out to a location (residency) of a patient living with learning disabilities. If found already unconscious, they are presented with the existing issues faced in Accident and Emergency departments across the UK, no access to medical records due to unknown variables. This was partially solved during the creation of the eHealth Passport application, as represented in Figure 29 with both a wireframe and the created search engine. A search engine was built to allow for manual QR code generation on a patient's device. This was made possible by allowing someone to search using a unique identifier created from 4 key pieces of information; forename, date of birth, house number, and postcode. On successful entry of this data, the QR code would appear on the patient's device without a geolocation check. However, this only solves a small fraction of emergency calls. What if the person was to collapse in the street? A paramedic would not know specific information about the patient, meaning QR code generation would still be delayed.

Testing

As with any software development project, bugs have arisen, which has slowed the development process whilst a solution was found. Generic manual testing was performed during the implementation phase with a further testing phase carried out to follow a test plan. Automated testing was used on major parts of the App Server to ensure, during new releases, vital aspects where not affected. This includes database connections, passport handling, and web services.

Table 5: Snapshot of test plan that was outlined to perform manual and automatic testing on the project. This was updated monthly to allow for any new additions to the project to be tested. Full version of the test plan can be seen in the Appendices.

Test	Module	Platform	Test Type	Conditional	Pass Criteria	Passing	Result
If no passport is provided, rejection page is displayed	Read Only	Web	White Box		404 Page is detected when the $_GET["passportID"] is omitted	Yes	Pass
If passport is provided, user can provide information and view readonly view	Read Only	Web	White Box		Forename and Surname can be provided when a correct $_GET["passportID"] is provided	Yes	Pass
Create User API works	API	Web	White Box		Sending the correct payload to the createuser api correctly creates a user in the db and returns the correct JSON format.	Yes	Pass
UserDetails API returns the data structure	API	Web	White Box		When a userID is sent to the userDetails API then JSON is returned and it matches the expected structure.	Yes	Pass
Can login via the API	API	Web	White Box		Encrypted username and password are accepted by the API and then, an Access Token with userID is returned	Yes	Pass
Can get the profile via the API	API	Web	White Box		Religion, Carer, Next of Kin, and Medical conditions are returned in JSON form and they all match the expected structure.	Yes	Pass
Can register an App ID via the API	API	Web	White Box		When device registration details are sent to the API, JSON is returned and matches the correct structure to say that the device was registered.	Yes	Pass

A test plan was created to ensure during each month, the mobile and web side of the eHealth Passport are still working technically. This test plan did not take into consideration the designs, but whether the eHealth passport was in fact functioning. Both white box and black box testing was performed on the eHealth passport to ensure both from a user and a developer point of view the application was functioning. For example, when logging into the health passport, black box testing was used to ensure the user was navigated correctly. Whereas, white box testing allows for the data to be checked to ensure it is correct and passed correctly to the mobile application when the app logs in.

The test plan was reviewed each month to ensure it was up to date to the current specification of the mobile application. Once the test plan was reviewed, a full test of

the mobile application and the App server was carried out with bug fixes performed quickly for monthly meetings. During the testing phase of the project, any serious limiting factors that were encountered during the process were logged.

Table 6: A table demonstrating critical issues that were faced and overcome during the development of the eHealth Passport Project. The table outlines both the problems that were faced and a brief explanation on how they were overcame.

Issue	**How it was overcome/fixed**
Https on iOS is blocking SSL	Utilising CFNetwork rather than NSUrlSession fixed this. Managed was out of the question due to TLS 1.0 no longer being used by major cloud developers since April 2018. This was actually not the case. TLS 1.0 is still being used by LJMU, I had to whitelist the domain
Https after install on iPad breaks CFNetwork.	Turns out SSL was not strong enough on LJMU Server. Without access, HTTP was granted on iOS devices to overcome
Forename surname check after encryption ALWAYS returns empty array as encryption obfuscates the values	Any encrypted values in the database that are required for a search are hashed so that the hashed value can be used to search again.
SSL error on Firebase Library	Removal of OpenSource Google Firebase library. Created a simple class that allows for data to be sent to devices via FCM using unique FCM tokens and cURL.
IP address is different if you are on WIFI or if you are on GSM	Rather than using IP address being used to uniquely identify devices, each device has a randomly generated UUID. This UUID is used to assign a user to an FCM token in the database over IP address. IP address is still logged in the database however.

Table 6 gives an idea about some of the issues faced during development. Some of the issues faced required large amounts of code being designed and rewritten. Some, however, required trivial fixes that would not be sufficient in industry but due to limiting factors had to temporarily be implemented. For example, the weak security on the CMPPROJ university server (TLS 1.0), meant a whitelist rule had to be made in the iOS version of the mobile application. This meant http had to be whitelisted in order for the mobile application to communicate with the App Server. Providing a vulnerability to the application. The ideal scenario would see the SSL certificates and TLS being updated to the latest version however, this could not be completed. This issue was also present on cURL calls on the App Server where a whitelist had to be implemented to allow for the application to talk to third party libraries like FCM.

Other issues, however, required a large amount of redesign and implementation to be completed again. Issues like IP Addresses being different on WiFi and GSM required a large amount of redesign, however, code changes were minimal. Originally, four factors of information were used to differentiate FCM tokens to assign them to a specific user. However, if the device moved between WiFi networks or GSM, it was likely the IP address (one of the four factors) would change. Causing a new FCM token to be issued. It was then designed to give each device a UUID (Universally Unique Identifier) to differentiate the devices. When the mobile application loads, a call is made to the App Server containing this UUID with the FCM token. If an update is required, this is then made. Once the user has then unlocked the mobile application or logged in, the FCM token on the App Server is updated to be alongside the corresponding user.

Table 7: Automated, black box, tests defined in the test plan. Codeception create the unit tests that are used to test the API and App Server. This is a snapshot of the full testing table seen in Appendix Two

Test	Module	Platform	Test Type	Conditional	Pass Criteria	Passing	Result
If no passport is provided, rejection page is displayed	Read Only	Web	White Box		404 Page is detected when the $_GET["passportID"] is omitted	Yes	Pass
If passport is provided, user can provide information and view readonly view	Read Only	Web	White Box		Forename and Surname can be provided when a correct $_GET["passportID"] is provided	Yes	Pass
Create User API works	API	Web	White Box		Sending the correct payload to the createuser api correctly creates a user in the db and returns the correct JSON format.	Yes	Pass
UserDetails API returns the data structure	API	Web	White Box		When a userID is sent to the userDetails API then JSON is returned and it matches the expected structure.	Yes	Pass
Can login via the API	API	Web	White Box		Encrypted username and password are accepted by the API and then, an Access Token with userID is returned	Yes	Pass
Can get the profile via the API	API	Web	White Box		Religion, Carer, Next of Kin, and Medical conditions are returned in JSON form and they all match the expected structure.	Yes	Pass
Can register an App ID via the API	API	Web	White Box		When device registration details are sent to the API, JSON is returned and matches the correct structure to say that the device was registered.	Yes	Pass

White box testing was also performed using an automated unit testing library; Codeception. This allowed for realistic behaviour to be simulated on the API via a command line interface meaning it was easy to determine whether specific code was not behaving as expected. Tests were written to login to the API, and to fetch data to ensure data was returned in the correct format, otherwise the app would not function correctly. Data types were checked to ensure they match the expected values. This also meant the API can be quickly tested during a new release onto the CMPPROJ server (from localhost) to ensure the database connections were still correct and incorrect configurations were not released. Table 7 displays all automated white box tests written using codeception for the API and the App Server. As seen, for basic automated testing, seven-unit tests were created to automatically test the project.

This allowed for the database to be tested to ensure the connection and interface was not broken during a release of a new version of the project.

Evaluation

This project saw the proposal, design, implementation, and testing of an elegant solution to the issues being faced with current paper-based Health Passports in the UK. The requirements originally devised, and refined were seen designed and implemented during the time frame of 6 months. During this time, time was expedited and hindered by unforeseen factors however, the project was still delivered on time with a working prototype seen earlier than expected.

This project was successfully created including clinical, geolocation and security aspects, which were required during the initial phases of the project. Implementation was successfully completed ahead of schedule with a proof of concept completed during a code sprint in January. This proof of concept sees a fully working health passport with full encryption (transport and rest), with a fully functioning geolocation delivery system.

However, although a successful project, some factors were reprioritised due to functionality. For example, the emergency services system required for Address and DOB to be implemented into the system. These factors help to uniquely identify a user in the system. However, they were originally seen as less of a priority to implement than that of other clinical information.

As seen in the requirements, Personal Care had a higher priority to be implemented over DOB and Address however, the requirements changed during the implementation phase to reprioritise them above personal care. Although still a high priority item, personal care was deemed the least priority out of all level 3 (number of occurrences) items. This is because both, Name and NHS number allow for a user to be uniquely identified within the NHS (so of a high priority) and carer, allows for further information to be found out about the patient – through a secondary source.

Table 8: Requirements specification displaying importance to implement against whether it was implemented in the project.

Element	Occurrence	Occurrence position	Implemented
Next of Kin	4	1	Yes
Name	3	1	Yes
NHS Number	3	1	Yes
Carer	3	1	Yes
Personal Care	3	1	No
Address	2	1	Yes
Phone	2	1	Yes
Medical history	2	1	Yes
Medication	2	1	No
DOB	2	1	Yes
GP Details	2	1	Yes
Allergies	2	1	Yes

Communication Needs	2	1	No
Risk/Safety	2	1	No
Seeing and Hearing	2	1	No
Eating and Drinking	2	1	No
How to take medications	2	1	No
How to go to the bathroom	2	1	No
Moving ability	2	1	No
Managing with pain	2	1	No
How to comfort	2	1	No
Sleeping routine	2	1	No
Level of support required	2	1	No
Things that will make stay better	2	1	No
Things that will make stay worst	2	1	No

Table 8 shows the requirements implemented alongside their importance. This is a snippet of the table seen earlier on in the Literature review of the current system, only elements with an occurrence of two or more are displayed in this table. As seen, ten out of twenty-four clinical elements were implemented into the project. As discussed previously, there were many backend implementations, which took priority over clinical data. Implementations like security and location services were a vital aspect of this application. These factors were vital due to the lack of security on SQLite on both Android and iOS and Location services, paired with reverse geocoding, being a vital aspect of the project.

Table 9:Displays all functional and non-functional requirements that were specified for the project.

Requirement	Achieved
Functional Requirements	
QR codes to appear on device lock screen when in healthcare centre (EG: Hospital),	Y
MySQL Database is used on Web Application,	Y
SQLite Database is embedded on the Mobile Application,	Y
Web Application is written in PHP,	Y
Mobile Application is written using Xamarin Forms (C#),	Y
Logged access is viewable by the owner of the health passport,	Y
When the user navigates away from the app, they are required to login again,	Y
Users are able to view a read,only view whenever they want,	Y
Anyone has the ability to scan the QR code without creating an account.	Y
QR Code directs the scanner to a web application.	Y

Data is secure during transmission and rest on App and Server.	Y
Users are able to provide their:	
Next of Kin	Y
Name	Y
NHS Number	Y
Carer	Y
Personal Care	N
Address	Y
Phone	Y
Medical History	Y
Medication	N
DOB	Y
GP Details	Y
Allergies	N
Communication Needs	N
Risk and Safety	N
Seeing and Hearing	N
Eating and Drinking	N
How to take Medications	N
How to go to the bathroom	N
Moving Ability	N
Managing with pain	N
How to comfort	N
Sleeping routine	N
Level of support required	N
Things that will make my stay better	N
Things that will make my stay worst	N
Basic information is provided and logged when someone views a read only version of a health passport:	
Address,	Y
Forename,	Y
Surname,	Y
Date/Time	Y
IP address	Y
Error logging to capture:	
File	Y
Directory	Y
Line	Y
Error Message	Y

Non,Functional Requirements	
The location of the device is gathered at least every 30 minutes.	Y
Secure authentication is used to determine whether the QR code has actually been scanned.	Y
The location is to be gathered whilst not significantly affecting battery life	Y
Code is fully documented and commented	Y
Screenshots cannot be taken when the mobile app is in full view	Y
Full fault detection on App server to log any errors	Y
Ability to dynamically alter the keywords for geolocation string matching	Y

Table 9 displays all requirements defined for this project, this table displays, which requirements were successfully implemented, and which are planned for future developments. As seen in the table, 52 requirements were originally defined as functional and non-functional. Out of these, 36 requirements were successfully fully designed, implemented and tested during the project with 16 postponed for future iterations during the software lifecycle. As seen in table 8 and 9, only clinical requirements were not fully implemented due to a lack of time. However, a fully operational eHealth Passport with geolocation and QR code access was achieved with the lack of these requirements being implemented.

During the testing phase of the application, testing was performed to measure the impact on a mobile phone's battery life the background task had. This was a large factor to consider when designing and implementing the project. As it was required to find a balance between the battery being affected and how accurately and often the location was gathered. In the requirements, it was defined that it was required to gather the location at least every thirty minutes. To satisfy this requirement, it was decided to start gathering the location every fifteen minutes and lower the frequency if the battery was affected too greatly.

At the end of each day (roughly after fifteen hours of average phone operation) battery usage was checked on a Pixel 3 device to see an estimate on the battery usage used by the eHealth Passport Android Application. It was planned to gather averages on a seven-day basis. If, the battery was seen to be greatly affected over a seven-day average, the test would be restarted with location gathered at fewer intervals (every twenty minutes for example). However, Table 10 demonstrates this was not needed. During the seven-day period it was seen on average, less than 1 per cent of the battery was needed for the eHealth Passport mobile application. With the battery usage being so low, it was deemed successful, for fifteen-minute intervals to be used to accurately judge the location of the device. This met the requirements defined early on during the development of this project and meant the user was not affected by having their battery drain at an alarming rate.

Table 10: Battery testing performed during a 7-day period.

Date	How Often Probed	Battery used for App
11th Jan	15 Minutes	Less than 1per cent Battery used for application
12th Jan	15 Minutes	Less than 1per cent Battery used for application
13th Jan	15 Minutes	Less than 1per cent Battery used for application
14th Jan	15 Minutes	Less than 1per cent Battery used for application
15th Jan	15 Minutes	Less than 1per cent Battery used for application
16th Jan	15 Minutes	Less than 1per cent Battery used for application
17th Jan	15 Minutes	Less than 1per cent Battery used for application
18th Jan	15 Minutes	Less than 1per cent Battery used for application

Currently without the keywords updating automatically, fine tuning of the Geocode algorithm is manual. This means, often, the reverse geocode is inaccurate due to the GPS detecting a different (but close) location. To determine the accuracy of the GPS and reverse geocode, a small API was written to store each detected location into the App Server's database. The position of each location is then plotted onto a Map with the use of the Google Map Roads API.

During a 2-hour period, the mobile device was left in a static location. The keyword 'Falkland' was added to the knowledge base to allow the notification to appear in the static location the device was kept in. However, the notification did not appear during any of the reverse geocodes. A mixture of both, the inaccuracy of GPS and an incomplete list of key phrases is seen to be the issue. Figure 30 shows the locations, which were discovered when attempting to display a notification within the 2-hour period.

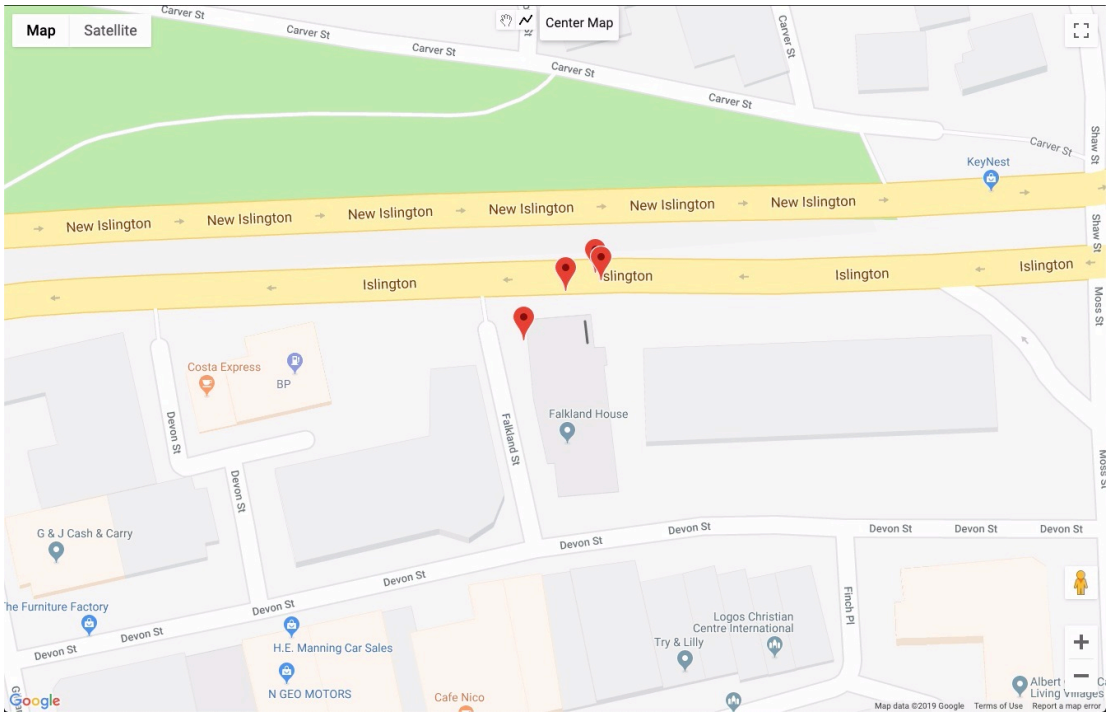


Figure 30: Map displaying all locations registered within a 2-hour period.

As seen in Figure 30, each location was given a red marker on the map during the 2-hour period. The area in which the mobile device was left has been marked in black on the map. As seen, although the markers are close to the device's location, they are not on the exact location. This means, an incorrect result will be given after the reverse geocode has been performed. Which, consequently, will mean the notification will not show. It is however, hoped once the location process starts to update keywords automatically, this issue will be remedied.

Project Planning

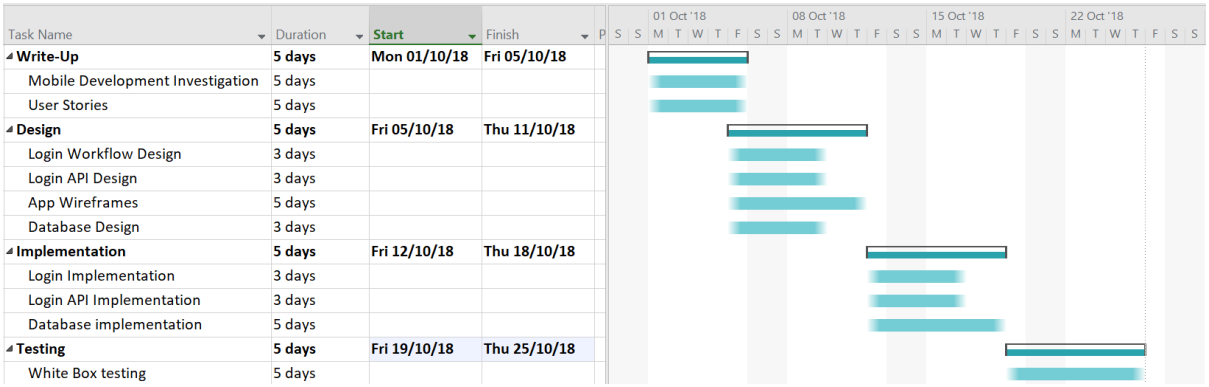
As discussed previously, an Agile methodology has been chosen to develop the eHealth Passport project. To ensure progress has been made, each month will be treated as a 'sprint'. A sprint is where there is a set amount of time to complete specific tasks (Scrummethodology.com, 2011). Each month will see work carried out that will be included within four categories; Writeup, Design, Implementation, and Testing. This can be defined as the following:

- Writeup: Any tasks, which involve writing documentation, literature, and/or requirements specification for the final project report.
- Design: Any tasks, which involve writing designs for the project. This could include Software UML, Wireframes, and Database design (entity relationship models).
- Implementation: Where any code is to be written based on designs from this or previous months.
- Testing: Where testing on the implementation is carried out to ensure it meets the design requirements.

At the start of each month, it will be required to plan what happens during each month sprint. This will be completed using Gantt charts in order to map a generic month workflow and to identify tasks, which are to be completed each month. This will allow for a schedule to be kept, allowing for the development to be kept on track. In the event that progress is not kept on track, due to unforeseen circumstances, these Gantt charts can help to define which areas still require work completing on the next month. They are also beneficial to see if a workload was too heavy, work was not completed in time due to sheer amount of work, was undertaken for one month, the workload can be lighter in future months to ensure this delay does not happen.

A monthly progress report will also be completed at the end of each sprint. This report will be used to explain in further detail what was completed during the month. This report will be used to demonstrate the work that has been completed alongside monthly meetings to give demonstrations on the progress made. Each Gantt chart sees rough guidelines on the timescales that will be taken to perform each action that is planned for the month. When a strict timescale has been set, blocks on the Gantt chart will be solid whereas, when a flexible timescale is provided (no dates, only duration) a blurred outline will be displayed on the time block. These were used for guidance to keep on track with the project, however, sometimes progress was made at a faster/slower timescale than anticipated.

October



Task Name	Duration	Start	Finish
Write-Up	**5 days**	**Mon 01/10/18**	**Fri 05/10/18**
Mobile Development Investigation	5 days		
User Stories	5 days		
Design	**5 days**	**Fri 05/10/18**	**Thu 11/10/18**
Login Workflow Design	3 days		
Login API Design	3 days		
App Wireframes	5 days		
Database Design	3 days		
Implementation	**5 days**	**Fri 12/10/18**	**Thu 18/10/18**
Login Implementation	3 days		
Login API Implementation	3 days		
Database implementation	5 days		
Testing	**5 days**	**Fri 19/10/18**	**Thu 25/10/18**
White Box testing	5 days		

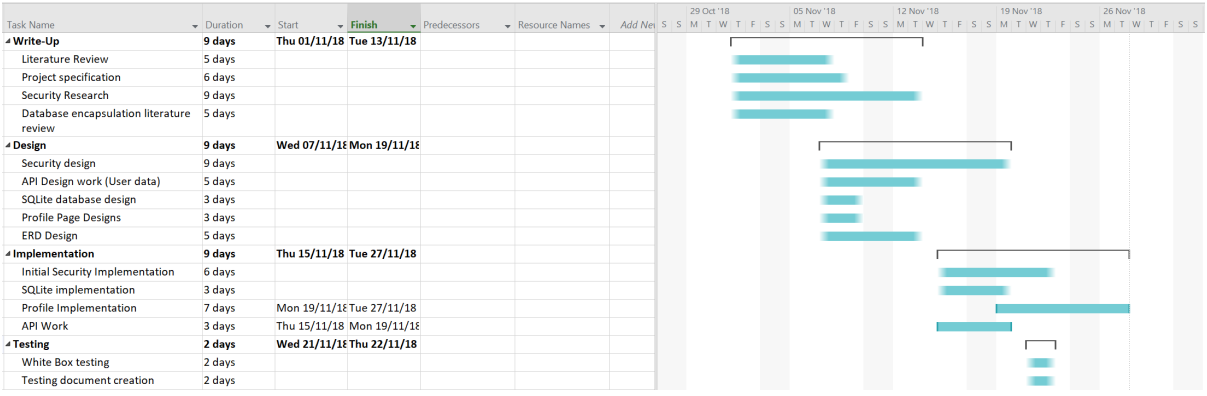
Figure 31: October's Gantt Chart outlining planned work during October

October saw the initial work completed on the Project. As seen in Figure 31, October saw initial investigations undertaken into the best course of action to develop the mobile language. This consisted of the best language to use to develop the application. With the result being Xamarin Forms; a Cross-Platform C# Library targeting both iOS and Android. User Stories were then created to help define designs during this month and further development sprints.

October's design and implementation phases saw the initial MySQL database creation with initial normalized ERD diagram created. This ERD, however, was constantly updated as the project grew. The Login was designed and implemented for the Mobile Application to use the OAuth 2.0 App Server that was designed and created.

Finally, Whitebox testing was performed to ensure the designs had been followed during the implementation phase.

November



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Write-Up	**9 days**	**Thu 01/11/18**	**Tue 13/11/18**		
Literature Review	5 days				
Project specification	6 days				
Security Research	9 days				
Database encapsulation literature review	5 days				
Design	**9 days**	**Wed 07/11/18**	**Mon 19/11/18**		
Security design	9 days				
API Design work (User data)	5 days				
SQLite database design	3 days				
Profile Page Designs	3 days				
ERD Design	5 days				
Implementation	**9 days**	**Thu 15/11/18**	**Tue 27/11/18**		
Initial Security Implementation	6 days				
SQLite implementation	3 days				
Profile Implementation	7 days	Mon 19/11/18	Tue 27/11/18		
API Work	3 days	Thu 15/11/18	Mon 19/11/18		
Testing	**2 days**	**Wed 21/11/18**	**Thu 22/11/18**		
White Box testing	2 days				
Testing document creation	2 days				

Figure 32: November's Gantt Chart outlining planned work during November

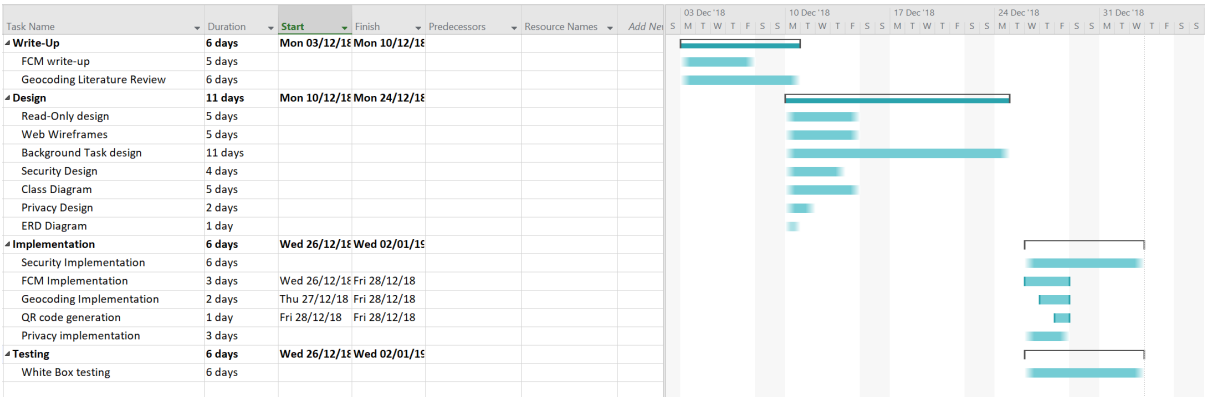
As seen in November's Gantt chart, stricter dates were given to design and implement certain aspects of the project. These were defined when a task could not be started until, a previous task was completed. A prime example of this would be the SQLite implementation and the Profile Implementation. This was in place as

without SQLite being in place on the Application, the profile could not be implemented due to the lack of storage.

As seen, November saw the writeup, design, implementation and testing of a large portion of groundwork for the application. Security considerations where made with regard to transmitting patient data between the App Server and Mobile Device.

White box was also accompanied with the creation of a testing document to log any issues that occurred during the creation of the mobile application.

December



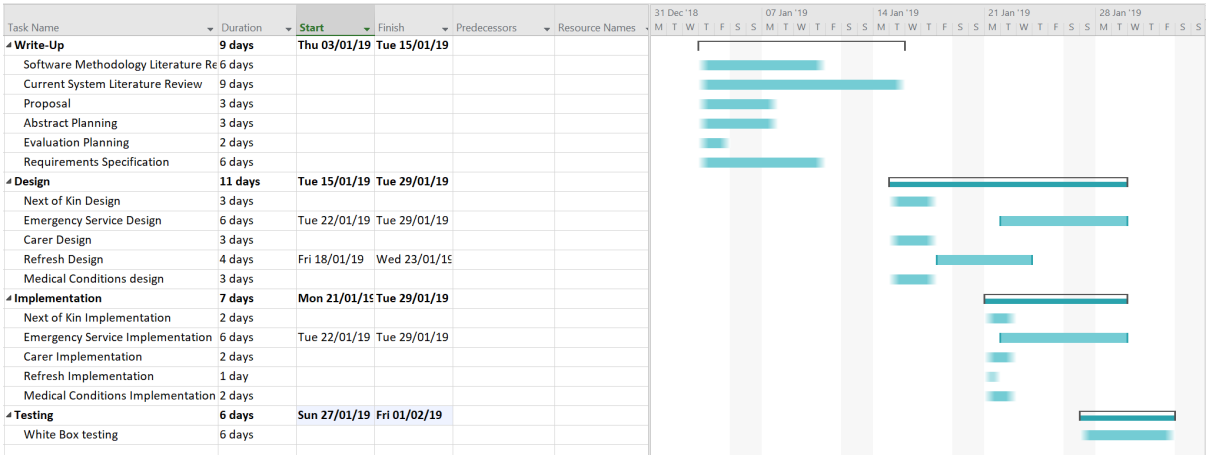
Task Name	Duration	Start	Finish	Predecessors	Resource Names
Write-Up	**6 days**	**Mon 03/12/18**	**Mon 10/12/18**		
FCM write-up	5 days				
Geocoding Literature Review	6 days				
Design	**11 days**	**Mon 10/12/18**	**Mon 24/12/18**		
Read-Only design	5 days				
Web Wireframes	5 days				
Background Task design	11 days				
Security Design	4 days				
Class Diagram	5 days				
Privacy Design	2 days				
ERD Diagram	1 day				
Implementation	**6 days**	**Wed 26/12/18**	**Wed 02/01/19**		
Security Implementation	6 days				
FCM Implementation	3 days	Wed 26/12/18	Fri 28/12/18		
Geocoding Implementation	2 days	Thu 27/12/18	Fri 28/12/18		
QR code generation	1 day	Fri 28/12/18	Fri 28/12/18		
Privacy implementation	3 days				
Testing	**6 days**	**Wed 26/12/18**	**Wed 02/01/19**		
White Box testing	6 days				

Figure 33: December's Gantt Chart outlining planned work during December

December saw a large amount of progress made on the Mobile Application with the design, implementation, and testing of background tasks, geolocation with reverse geocoding and generation of QR codes. This saw a proof of concept made on delivering the eHealth Passport via a location based QR code medium. Seeing a large amount of design and implementation with strict start times, as often, tasks had to be completed sequentially. For example, during the implementation of FCM, the geolocation and geocoding was required in order to display the notification, which required the QR code.

A large effort was made in designs during this month, this was because of issues faced with background tasks that were discussed in the report. The move was made to FCM during the implementation phase to implement a background task.

January



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Write-Up	**9 days**	**Thu 03/01/19**	**Tue 15/01/19**		
Software Methodology Literature Re	6 days				
Current System Literature Review	9 days				
Proposal	3 days				
Abstract Planning	3 days				
Evaluation Planning	2 days				
Requirements Specification	6 days				
Design	**11 days**	**Tue 15/01/19**	**Tue 29/01/19**		
Next of Kin Design	3 days				
Emergency Service Design	6 days	Tue 22/01/19	Tue 29/01/19		
Carer Design	3 days				
Refresh Design	4 days	Fri 18/01/19	Wed 23/01/19		
Medical Conditions design	3 days				
Implementation	**7 days**	**Mon 21/01/19**	**Tue 29/01/19**		
Next of Kin Implementation	2 days				
Emergency Service Implementation	6 days	Tue 22/01/19	Tue 29/01/19		
Carer Implementation	2 days				
Refresh Implementation	1 day				
Medical Conditions Implementation	2 days				
Testing	**6 days**	**Sun 27/01/19**	**Fri 01/02/19**		
White Box testing	6 days				

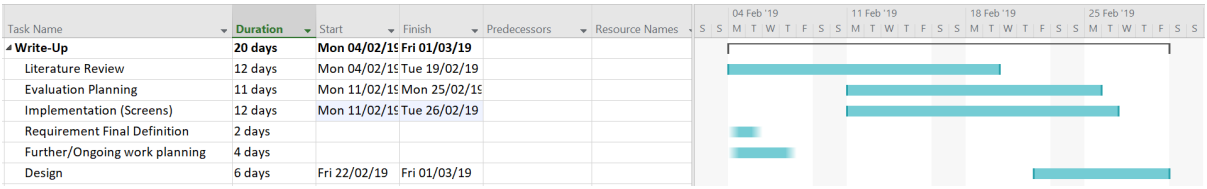
Figure 34: January's Gantt Chart outlining planned work during January

January saw the final push for implementation during the project. To reach a full proof of concept with minimum requirements (due to project timescales), Next of Kin, Carer, Medical Conditions, and Emergency Service access was implemented into the App Server and Mobile Application. This saw large amounts of design, implementation, and testing undertaken in order to reach the month's workload target.

January also saw a large push for writeup to be completed. The literature review for the project was completed with a full proposal written outlining what will be delivered during this project. Lastly, a full list of requirements was written up, although there were vague requirements based on the technical requirements of the project, no non-functional requirements had been defined until this point.

White box testing was once again performed to ensure no issues occurred during the implementation phase and the implementation correctly matched the designs created.

February



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Write-Up	**20 days**	**Mon 04/02/19**	**Fri 01/03/19**		
Literature Review	12 days	Mon 04/02/19	Tue 19/02/19		
Evaluation Planning	11 days	Mon 11/02/19	Mon 25/02/19		
Implementation (Screens)	12 days	Mon 11/02/19	Tue 26/02/19		
Requirement Final Definition	2 days				
Further/Ongoing work planning	4 days				
Design	6 days	Fri 22/02/19	Fri 01/03/19		

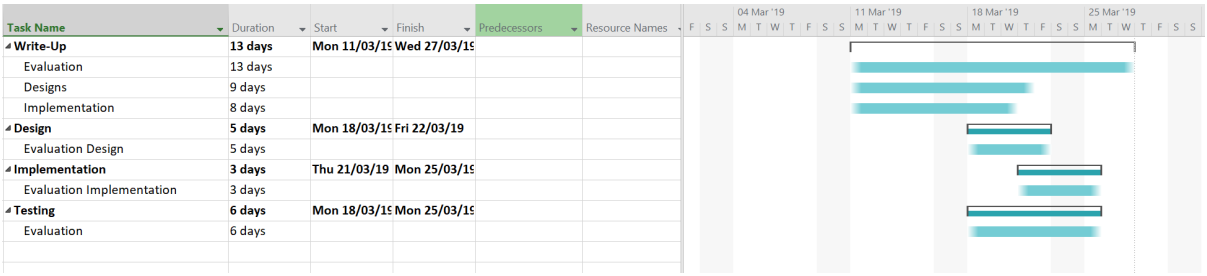
Figure 35: February's Gantt Chart outlining planned work during February

With a large amount of design and implementation conducted in the previous month, the decision was made to focus on writeup aspects of the report. This allowed for the literature review to be finalised with all implementation that had been conducted in the previous months to be written up.

As it was coming to the realisation how large this project was, a further/ongoing work section was started to be planned to outline work that would be completed in the future.

Finally, the design section had further work completed to writeup any remaining wireframes and database designs.

March



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Write-Up	**13 days**	**Mon 11/03/19**	**Wed 27/03/19**		
Evaluation	13 days				
Designs	9 days				
Implementation	8 days				
Design	**5 days**	**Mon 18/03/19**	**Fri 22/03/19**		
Evaluation Design	5 days				
Implementation	**3 days**	**Thu 21/03/19**	**Mon 25/03/19**		
Evaluation Implementation	3 days				
Testing	**6 days**	**Mon 18/03/19**	**Mon 25/03/19**		
Evaluation	6 days				

Figure 36: March's Gantt Chart outlining planned work during March

During the final sprint in March. The focus was placed onto the evaluation of the project. A small program was designed and implemented to track how accurate the geolocation and reverse geocode was at finding the device's location. This program was then used to help evaluate the project.

Ongoing work

As this project did not see all requirements fully designed and implemented, it is expected to carry on the development of this project to fully create the eHealth Passport. With this improvement, new wireframes can be designed to allow for the most efficient designs to be made for certain pages. For example, pages gathering information such as Personal Care will require one large section of text for a patient to provide information. A more suitable wireframe design should be drawn to accommodate for this in the future.

A design, which sees a large body of text visible, and editable by a user, which can be saved and sent to the App Server is more suitable for this. It will also be required to design how this is displayed in the read only view as large bodies of text have not yet been accounted for. Although wireframes will need to be designed, the App Server is already designed to handle new data being added. It would be required for a new table to be added to the database with a method designed to fetch the data and return it. With the object-oriented approach used when creating the App Server, this is easily completed.

Interoperability is another vital improvement required for the adoption of this project. SNOMED CT, a clinical terminology dictionary (NHS Digital, 2019), is required by the NHS for integrations by 2020 (NHS Digital, 2019). To successfully implement SNOMED CT, HL7 FHIR can be used. HL7 FHIR is an interoperability standard, which can be used to increase clinical value of a health system (Bender and Sartipi, 2013). The use of the interoperability standard FHIR, will allow for the health passport to be implemented into any existing health system that supports it. The NHS, a large target for this project, has recently released a new API standard

allowing for third-party integrations via a FHIR interoperable API. Therefore, it is possible, with the use of FHIR, to get accurate real patient data from a live database.

Periodically, users have registered for a health passport can have their live patient data pulled from the NHS server. This would be possible with the use of an NHS number provided by the user and the health passport can be populated. Minimal changes would be required to get this to integrate successfully. An endpoint would be created in the API to allow for FHIR data to be sent from the NHS API to the health passport's app server. Then, a new CRON job would be created to periodically check for any new data for the API to fetch. This can then be used to update the MySQL database for the health passport, which will allow for both the Mobile Application to edit the data, and the Read Only view to display the data.

Conclusion

This project saw the successful adaptation of existing paper-based health passports. The project oversaw the development of a Cross-Platform mobile application to store and hold a health passport. Alongside this, an app server was created to have a central location to store and hold user data. This allowed for multiple devices, and users to have access to health passports.

To overcome the issues of outdated methods of delivery, a new strategy of delivering patient records was created. The use of location based QR codes allows for a user to scan and have quick access to a specific patient's records. With the use of GPS and reverse geocoding, it is possible to accurately determine whether a device is within the radius of a healthcare facility, and using this information, display a QR code within a notification on the device's lock screen. With the use of this QR code, it is possible for health care professionals to see limited access to a specific health passport.

The use of partial string matching has allowed for a reverse geocoded address to be checked for keywords. Keywords are sent from the app server to mobile devices each time the location is derived, which allows for keywords to be refined and tuned to allow for location to be made more accurate as time goes on.

An investigation was conducted into how much of an impact the battery life of android devices had with location being gathered at set intervals. As seen, less than 1 per cent of battery – over the course of a seven-day period was used by the mobile application. This means, the mobile application is able to successfully derive a devices location and deliver a health passport to a health professional without negatively affecting user experience.

References

Abbott, A., (2014) Communication and Continuity in Progressive , Life-Limiting Illness. [online] January. Available at: http://www.rcgp.org.uk/rcgp-near-you/rcgp-northern-ireland/~/media/Files/RCGP-Faculties/Northern-Ireland/RCGP-Healthcare-Communication-report.ashx.Alborz,

A., Mcnally, R. and Glendinning, C., (2005) *Access to health care for people with learning disabilities in the UK: mapping the issues and reviewing the evidence*. *Journal of Health Services Research & Policy*.

Agiledata.org. (n.d.). *Encapsulating Database Access: An Agile 'Best Practice'*. [online] Available at: http://www.agiledata.org/essays/implementationStrategies.html [Accessed 25 Apr. 2019].

Ali, A., Scior, K., Ratti, V., Strydom, A., King, M. and Hassiotis, A., (2013) Discrimination and Other Barriers to Accessing Health Care : Perspectives of Patients with Mild and Moderate Intellectual Disability and Their Carers. 88.

Android Developers. (n.d.). *Background Execution Limits | Android Developers.* [online] Available at: https://developer.android.com/about/versions/oreo/background [Accessed 16 Feb. 2019].

Art, P., (2010) Equality Act 2010, Chapter 15. [online] 20101, p.15. Available at: http://www.legislation.gov.uk/ukpga/2010/15/contents.

Balaji, S., (2012) Waterfall vs V-Model vs. Agile: A Comparative Study on SDLC [Electronic version]. *International Journal of Information Technology and Business Management*, [online] 21, pp.26–30. Available at: http://jitbm.com/Volume2No1/waterfall.pdf.

Bender, D. and Sartipi, K., (2013) HL7 FHIR: An agile and RESTful approach to healthcare information exchange. *Proceedings of CBMS 2013 - 26th IEEE International Symposium on Computer-Based Medical Systems*, pp.326–331.

Docs.microsoft.com. (n.d.). Xamarin.Essentials: Geocoding - Xamarin. [online] Available at: https://docs.microsoft.com/en-us/xamarin/essentials/geocoding?context=xamarin%2Fxamarin-forms&tabs=android [Accessed 4 Dec. 2018].

Docs.microsoft.com. (n.d.). *Xamarin.Forms - Xamarin.* [online] Available at: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/ [Accessed 25 Apr. 2019].

Emerson, E. and Baines, S., (2011) Health inequalities and people with learning disabilities in the UK. *Tizard Learning Disability Review*.HMSO, (1990) Computer Misuse Act 1990 ARRANGEMENT OF SECTIONS. [online] pp.2–12. Available at: http://www.legislation.gov.uk/ukpga/1990/18/pdfs/ukpga_19900018_en.pdf.

Fontawesome.com. (2019). *Font Awesome 5. [online]* Available at: https://fontawesome.com/ [Accessed 15 Apr. 2019].

Google Developers. (n.d.). *Get Started | Geocoding API | Google Developers.* [online] Available at: https://developers.google.com/maps/documentation/geocoding/start [Accessed 25 Apr. 2019].

Houghton, B.-M., (2001) Caring for people with Down syndrome in A&E.

Ico.org.uk. (n.d.). *Passwords in online services. [online]* Available at: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/security/passwords-in-online-services/ [Accessed 25 Apr. 2019].

Imam, K., (2018) *Personal Health Card: Use of QR Code to Access Medical Data*. [online] Available at: http://digitalcommons.uri.edu/theseshttp://digitalcommons.uri.edu/theses/1198.

Kwon, H.S., Cho, J.H., Kim, H.S., Lee, J.H., Song, B.R., Oh, J.A., Han, J.H., Kim, H.S., Cha, B.Y., Lee, K.W., Son, H.Y., Kang, S.K., Lee, W.C. and Yoon, K.H., (2004) Development of web-based diabetic patient management system using short message service (SMS). In: *Diabetes Research and Clinical Practice*.

Lee, J., (2012) *Health Passport Guidance notes for completion Background information*.

Macarthur, J., Brown, M., Mckechanie, A., Mack, S., Hayes, M. and Fletcher, J., (2015) Making reasonable and achievable adjustments: The contributions of learning disability liaison nurses in 'Getting it right' for people with learning disabilities receiving general hospitals care. *Journal of Advanced Nursing*.

Mencap. (n.d.). *Learning disability and conditions*. [online] Available at: https://www.mencap.org.uk/learning-disability-explained/learning-disability-and-conditions [Accessed 25 Apr. 2019].

McMurray, A. and Beebee, J., (2007) *Learning disability awareness training for hospital staff*.National Patient Safety Agency, (2004) Understanding the patient safety issues for people with learning disabilities. [online] Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Understanding+the+patient+safety+issues+for+people+with+learning+disabilities#0.

Microsoft. 2019. Visual Studio (8.2.0). [Software]. 20th April 2019.

NHS Digital. (2019). *SNOMED CT - NHS Digital.* [online] Available at: https://digital.nhs.uk/services/terminology-and-classifications/snomed-ct [Accessed 26 Apr. 2019].

Owen, S. (2019). *zxing/zxing*. [online] GitHub. Available at: https://github.com/zxing/zxing [Accessed 10 Apr. 2019].

Pressman, R. (2015). *Software engineering*. Boston: McGraw-Hill Education.

Poushter, J., (2016) Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies: But advanced economies still have higher rates of technology use. *Pew Research Center*, [online] pp.1–5. Available at: http://www.pewglobal.org/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/.

RCGPNI, (2014) What I Need You to Know: A Health and Care Record for Me, My Family and Carers - Guidance Notes. [online] Available at: http://www.rcgp.org.uk/rcgp-near-you/rcgp-northern-ireland/~/media/Files/RCGP-Faculties/Northern-Ireland/RCGP-Healthcare-passport-guidance.ashx.

Record, C., Family, M. and Version, P., (n.d.) My Healthcare Passport My Healthcare Passport What I Need You to Know A Health and Care Record for Me, My Family and Carers My Healthcare Passport ©.S., L., J., O. and C., M., (2017) Effective? Engaging? Secure? Applying the ORCHA-24 framework to evaluate apps for chronic insomnia disorder. *Evidence-based mental health*, [online] 204, p.e20. Available at: http://www.embase.com/search/results?subaction=viewrecord&from=export&id=L623331330%0Ahttp://dx.doi.org/10.1136/eb-2017-102751.

Schaub, F., Deyhle, R. and Weber, M., (2012) Password Entry Usability and Shoulder Surfing Susceptibility on Different Smartphone Platforms.

Scrummethodology.com. (2011). Scrum Sprint. [online] Available at: http://scrummethodology.com/scrum-sprint/ [Accessed 25 Apr. 2019].

Talkback-uk.com. (n.d.). *Health Passport - Talkback.* [online] Available at: http://www.talkback-uk.com/our-work/health-passport [Accessed 25 Apr. 2019].

Appendices

Appendix One: Unfiltered table of elements from Health Passports

Element	Occurrence	No of Occurrence
Next of Kin	4	1
Name	3	1
NHS Number	3	1
Carer	3	1
Personal Care	3	1
Address	2	1
Phone	2	1
Medical history	2	1
Medication	2	1
DOB	2	1
GP Details	2	1
Allergies	2	1
Communication Needs	2	1
Next of Kin	4	2
Risk/Safety	2	1
Seeing and Hearing	2	1
Eating and Drinking	2	1
Name	3	2
How to take medications	2	1
Address	2	2
How to go to the bathroom	2	1
NHS Number	3	2
Moving ability	2	1
Next of Kin	4	3
Carer	3	2
Managing with pain	2	1
How to comfort	2	1
Sleeping routine	2	1
Level of support required	2	1
Things that will make stay better	2	1
Things that will make stay worst	2	1
Medical history	2	2
All about me	1	1
Tasks able to complete with support	1	1
What my carer wants you to know	1	1
Risks	1	1

Medical Condition	1	1
Useful information	1	1
Notes	1	1
Date completed	1	1
Mental capacity assessment	1	1
Social Services number	1	1
Power of attorney	1	1
Spirituality	1	1
Disabilities and Impairments	1	1
Medications	1	1
Discharge arrangements	1	1
I am happy when	1	1
I am sad when	1	1
Dressing and undressing	1	1
Sitting	1	1
Standing	1	1
Mobility aid?	1	1
Personal care	3	2
Brushing my teeth	1	1
Bathing and washing	1	1
Name	3	3
Phone	2	2
DOB	2	2
NHS Number	3	3
GP Details	2	2
Religion	1	1
Next of Kin	4	4
Carer	3	3
Medication	2	2
Allergies	2	2
Medical Conditions	1	1
Consent	1	1
Communication Needs	2	2
Behaviour	1	1
Risk/Safety	2	2
Seeing and Hearing	2	2
Eating and Drinking	2	2
How to take medications	2	2
How to go to the bathroom	2	2

Moving ability	2	2
Managing with pain	2	2
How to comfort	2	2
Sleeping routine	2	2
Personal Care	3	3
Level of support required	2	2
Things that will make stay better	2	2
Things that will make stay worst	2	2
Accessible version	1	1

Appendix Two: Full automated and manual test plan that was used to test during development

#	Test	Module	Platform	Test Type	Conditional	Passing	Result
1	Correct credentials logs user in	Login	App	Black Box		Yes	Pass
2	Incorrect credentials rejects login	Login	App	Black Box		Yes	Pass
3	Nothing provided displays login error	Login	App	Black Box		Yes	Pass
4	username cannot be null	Registering	App	Black Box		Yes	Pass
5	username cannot already exist	Registering	App	Black Box		Yes	Pass
6	email address cannot be null	Registering	App	Black Box		Yes	Pass
7	password cannot be null	Registering	App	Black Box		Yes	Pass
8	forename cannot be null	Registering	App	Black Box		Yes	Pass
9	surname cannot be null	Registering	App	Black Box		Yes	Pass
10	address cannot be null	Registering	App	Black Box		Yes	Pass
11	postcode cannot be null	Registering	App	Black Box		Yes	Pass
12	user successfully registers	Registering	App	Black Box		Yes	Pass
13	username automatically populates	Lock	App	Black Box		Yes	Pass
14	user is successfully logged in with correct password	Lock	App	Black Box		Yes	Pass

15	user is not logged in with incorrect password	Lock	App	Black Box		Yes	Pass
16	Building message appears on successful login	Login	App	Black Box		Yes	Pass
17	Building message appears on successful login	Lock	App	Black Box		Yes	Pass
18	Name is populated correctly (preferred or forename)	Dashboard	App	Black Box		Yes	Pass
19	Passport viewed successfully populates	Dashboard	App	Black Box		Yes	Pass
20	Buttons work	Dashboard	App	Black Box		Yes	Pass
21	Navigation works	All	App	Black Box		Yes	Pass
22	Logout button works	All	App	Black Box		Yes	Pass
23	Profile picture appears	Profile	App	Black Box		Yes	Pass
24	Forename is populated	Profile	App	Black Box		Yes	Pass
25	Middle name is populated	Profile	App	Black Box	Y	Yes	Pass
26	DOB is populated	Profile	App	Black Box		Yes	Pass
27	Gender is populated	Profile	App	Black Box	Y	Yes	Pass
28	NHS Number is populated	Profile	App	Black Box	Y	Yes	Pass
29	Preferred name is populated	Profile	App	Black Box	Y	Yes	Pass
30	Phone number is populated	Profile	App	Black Box	Y	Yes	Pass
31	Email address is populated	Profile	App	Black Box		Yes	Pass
32	Religion is populated	Profile	App	Black Box	Y	Yes	Pass
33	Next of kin is populated	Profile	App	Black Box	Y	Yes	Pass
34	Carer is populated	Profile	App	Black Box	Y	Yes	Pass
35	Correct medical conditions are selected	Profile	App	Black Box	Y	Yes	Pass
36	Data is updated successfully	Profile	App	Black Box		Yes	Pass
37	Navigating away from the app displays lock screen	All	App	Black Box		Yes	Pass
38	Correct data is displayed for people who have viewed passport	Privacy	App	Black Box	Y	Yes	Pass
39	If an address has not been provided a message is displayed to say this	Privacy	App	Black Box		Yes	Pass

40	QR code is generated successful	Settings	App	Black Box		Yes	Pass
41	QR code scans successfully	Settings	App	Black Box		Yes	Pass
42	QR code directs user to the correct passport	Settings	App	Black Box		Yes	Pass
43	View passport button directs user to the correct passport	Settings	App	Black Box		Yes	Pass
44	Base information is always displayed	Read Only	Web	Black Box		Yes	Pass
45	If something in base information is null, "Not Provided" is displayed	Read Only	Web	Black Box	Y	Yes	Pass
46	If Next of Kin is provided, it is displayed	Read Only	Web	Black Box		Yes	Pass
47	If Next of Kin is not provided, it is hidden	Read Only	Web	Black Box		Yes	Pass
48	If Carer is provided, it is displayed	Read Only	Web	Black Box		Yes	Pass
49	If carer is not provided it is hidden	Read Only	Web	Black Box		Yes	Pass
50	If a medical condition is provided, it is displayed in a bullet list	Read Only	Web	Black Box		Yes	Pass
51	Not provided is displayed for all visible null fields	Read Only	Web	Black Box		Yes	Pass
52	If user exists, notification is sent	Emergency Access	Web	Black Box		Yes	Pass
53	If user does not exist, error is displayed	Emergency Access	Web	Black Box		Yes	Pass
54	Notifications successfully appear	Notification	App	Black Box		Yes	Pass
55	Notification can be scanned	Notification	App	Black Box		Yes	Pass
56	Notification QR code directs the scanner to the correct passport	Notification	App	Black Box		Yes	Pass
57	Forename can be entered	Emergency Access	Web	Black Box		Yes	Pass
58	Surname can be entered	Emergency Access	Web	Black Box		Yes	Pass
59	If captcha is incorrect, passport is not displayed	Read Only	Web	Black Box		Yes	Pass
60	If forename is empty, passport is not displayed	Read Only	Web	Black Box		Yes	Pass
61	If surname is empty, passport is not displayed	Read Only	Web	Black Box		Yes	Pass
62	If terms and conditions are not accepted, passport is not displayed	Read Only	Web	Black Box		Yes	Pass
63	If a passport that does not exist is requested, user is rejected	Read Only	Web	Black Box		Yes	Pass

64	If no passport is provided, user is rejected	Read Only	Web	Black Box		Yes	Pass
65	If emergency access is used, location is not checked	Notification	App	Black Box		Yes	Pass
66	If no passport is provided, rejection page is displayed	Read Only	Web	White Box		Yes	Pass
67	If passport is provided, user can provide information and view readonly view	Read Only	Web	White Box		Yes	Pass
68	Create User API works	API	Web	White Box		Yes	Pass
69	UserDetails API returns the data structure	API	Web	White Box		Yes	Pass
70	Can login via the API	API	Web	White Box		Yes	Pass
71	Can get the profile via the API	API	Web	White Box		Yes	Pass
72	Can register an App ID via the API	API	Web	White Box		Yes	Pass

Appendix Three: Monthly Meeting Repository

November

BSc(Hons) Final Year Project
Monthly Supervision Meeting Record

Student: .....Matthew Nethercott
..... Date: November

Main issues / Points of discussion / Progress made
Xamarin Forms chosen for development. Initial Wireframes designed for the App UI. Initial User Stories gathered for requirements. Initial Database designs completed. Implementation started with initial database design implemented, Login workflow implemented on Xamarin and Web app. Authentication of requests to API and started work on API calls however they are not yet FHIR compliant.
Actions for the next month
Wireframes designed for Web platform. Literature review complete for choice of Geolocator API. Initial API calls created will be moved to a FHIR compliant structure. Research completed on securing data during transit between the App and the Web Platform. Further implementation work completed on web and app.
Deliverables for next time
-Literature review report -Specification document

Other comments
Very good progress so far.

Supervisor signature: Prof. Abdennour El Rhalibi
Student signature: MNethercott .....

December

BSc(Hons) Final Year Project
Monthly Supervision Meeting Record

Student: .....Matthew Nethercott
..... Date: 12th December

Main issues / Points of discussion / Progress made
Literature review started for Geolocating API however, paused for December for development on Location services to start so final choices can be made. Database Encapsulation vs Database calls Literature review written. Security Design started. Sequence diagram has been created to demonstrate early prototype ideas for security design. Research into whether HTTPS is sufficient in itself as a standalone security protocol has been started. Project specification started to the IEEE 830 standard written specifically for Software Requirements Specifications. Section 2 from this guideline has been completed. Section 2 provides initial high-level outlines for project. A start has been made on Section 4 (Design). Further development has been made on the Web application and Mobile application. User data has been created in the database and is being fetched from an API call. An embedded database has been created for the mobile application which mirrors the structure of the main application however is user specific. This has been created SQLite and is populated upon login. Code redundancies have been removed (unnecessary code has been trimmed to make the application more efficient). This is now being displayed in a profile page on the mobile application. Base64 has been chosen for sending images over the web in a JSON format. A testing document has been started to gather a log of what has been tested and any issues found in the application. Issues and how they are overcome are being tracked in an online spreadsheet for discussion during testing phase and evaluation phase of the document. Bug fixes and issues have been fixed on iOS ensuring both target devices now support the application from the same codebase. An ERD diagram has been created although this will be updated due to Agile Development Methodology.
Actions for the next month
Further work to be completed on IEEE documentation. Location services and background tasks to be created for the application over Christmas break. Extensive testing to be carried out to maximise efficiency for maintaining battery life. QR Code generation on App with Endpoint created for read-only view on web application. Class diagram created for current state of mobile application.

Although defined for last months work, an investigation is needed for FHIR compliance. This should be designed over the Christmas break and decided whether to continue to aim for FHIR compliance.
Deliverables for next time
The progress are very good so far. The design, development and documentation carried out and proposed for next month are suitable.
Other comments

Supervisor signature: Prof. Abdennour El Rhalibi

Student signature: MNethercott .....

January

BSc(Hons) Final Year Project
Monthly Supervision Meeting Record

Student: .....Matthew Nethercott
..... Date: January

Main issues / Points of discussion / Progress made
Read Only view endpoint has been created which gathers data from the user who is viewing the passport. With wireframes created to ensure design is consistent with Mobile Application. Push notifications have been implemented and tested on Android however, they have not been tested on iOS due to the lack of a paid apple developer account. Created an integration with Firebase Cloud Messaging (FCM) to allow the App Server to send data to targeted devices. This runs automatically at periodic times throughout the day. Geocoding Literature review has been written to help decide which API to use when calculating whether the device is situated within a healthcare facility. String matching/searching has been used to detect whether keywords are present within a JSON object sent from the chosen API. Security considerations have been written and implemented into both the Web Application and the Mobile Apps. Class diagram created for the Mobile Application to the current state of the application. ERD Diagram updated to current state of the application.
Actions for the next month
Further Medical data to be implemented into the mobile application and read only view. Refreshing of data when updated on App Server to be implemented on the Mobile Application (Who viewed passport). Further testing to be done on the mobile application and read only view. Further work on write-up with written work on design considerations, implementation, and testing already carried out to be added.
Deliverables for next time
The project is going very well, and you are doing very good progress with the development.

As discussed, you may want to start to document the background, literature review and the requirements specification. The application development is progressing very well, you may want to start thinking about how to test and evaluate the system and application.
Other comments
I am happy with the progress so far.

Supervisor signature:

Prof. Abdennour El Rhalibi

Student signature: MNethercott .....

February

BSc(Hons) Final Year Project
Monthly Supervision Meeting Record

Student: .....Matthew Nethercott
..... Date: February

Main issues / Points of discussion / Progress made
Further implementation progress made: Refreshing data, medical conditions, next of kin, carer, emergency services portal (issue found and created from issue) and registering users. (both on app and read only portal) Background researched and written. Literature review for software methodology written. Literature review into previous health passport uses started. Proposal written. Requirements specification defined from previous health passports. Initial plans on how to evaluate the project have been gathered. Started thinking of points to discuss in the Abstract.
Actions for the next month
Evaluation of project. Evaluation of Software Product. Final steps of implementation including: Medications and Allergies. Continue ongoing work on the report. Refining and improving the report as I go. Including Abstract, Implementation, Finalising Design.
Deliverables for next time
First draft of the project report.
Other comments
Very good progress with the specification, design and implementation

Supervisor signature: Prof. Abdennour El Rhalibi

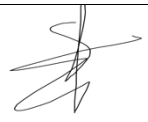
Student signature: MNethercott .....

March

BSc(Hons) Final Year Project
Monthly Supervision Meeting Record

Student: .....Matthew Nethercott
..... Date: 28th March

Main issues / Points of discussion / Progress made
Work completed on final project report • Evaluation started • Implementation • Designs • Requirements • Literature review • Further/Ongoing work stubbed out
Actions for the next month
Finish project write-up and prepare presentation.
Deliverables for next time
Draft of complete project report for feedback. Demos of the application
Other comments



Supervisor signature: Prof. Abdennour El Rhalibi.....
Student signature: MNethercott .....

Appendix Four: Code and Users

Code: https://ljmu-my.sharepoint.com/:f:/g/personal/cmpmneth_ljmu_ac_uk/Ej1he3izkMpEsx8z7i2gg6kBx3P_W7DFF0vsSvzSrGi7hA?e=5kMRXU

Read Only Portal: http://cmpproj.cms.livjm.ac.uk/cmpmneth/webplatform/Login/passportEndpoint.php?passportID=fl8Gl75dNe

Emergency Access Portal: https://cmpproj.cms.livjm.ac.uk/cmpmneth/webplatform/

Username: ljmureview
Password: testuser1