

FACULTY OF SCIENCE, ENGINEERING AND COMPUTING

School of *Computer Science & Mathematics*

**BSc DEGREE
IN**
Computer Science

PROJECT REPORT

Name: Suvarkka Rajendram

ID Number: K1554050

Project Title: Healthcare Management System

Project Type: Build

Date: 23/04/2018

Supervisor: Graeme Jones

Kingston University London

Plagiarism Declaration

The following declaration should be signed and dated and inserted directly after the title page of your report:

Declaration

I have read and understood the University regulations on plagiarism and I understand the meaning of the word *plagiarism*. I declare that this report is entirely my own work. Any other sources are duly acknowledged and referenced according to the requirements of the School of Computer Science and Mathematics. All verbatim citations are indicated by double quotation marks ("..."). Neither in part nor in its entirety have I made use of another student's work and pretended that it is my own. I have not asked anybody to contribute to this project in the form of code, text or drawings. I did not allow and will not allow anyone to copy my work with the intention of presenting it as their own work.

Date 23/04/2018

Signature Suvarkka Rajendram

Abstract

The advancements in technology have led to the digitization of the healthcare industry. Several healthcare organisations now use a database system which allows them to store and manage huge amounts of data. This can be accessed by a wide range of users where they are also able to share this data with each other. However, these types of system are expensive to purchase and some only offer limited services. A healthcare management system will be created for this project and will be developed for smaller healthcare practices such as GP surgeries that do not have an electronic system in place. This system will provide an essential appointment service as well as clinicians will be able to have access to other features which allows them to be more efficient in providing care. In addition, patients will have access to local GP services online which will give them more flexibility. Importance will be made on this project to produce a user friendly, responsive system.

Introduction – Chapter 1

1.1 Background

A GP practice usually have a team which includes doctors, nurses and administrative staff. The clinicians deal with a range of health problems, give out vaccinations and perform simple surgical operations. Patients can access these services by making appointments and having consultations with either a doctor or a nurse. They give patients relevant information or prescribe medications depending on the patient's diagnosis and medical history. These consultations also can be done by over the phone. In addition, reminders get sent to patients about vaccinations that are due soon where it will inform them to make appointments. Importantly, doctors and nurses are able to provide continuity of care to patients of all ages.

1.2 Problem and Solution

GP surgeries that are newly opened may not have an electronic system to record and manage their data. It would also not be ideal for them to use a paper-based system for storing information. This is because it would require extensive paper work and a considerable amount of time to update or make changes to data such as patient details. Also, it can be difficult when searching for a specific record and may not be available to more than one user. This can be considered as inefficient and insecure as documents may also be missed. The system that will be developed for this project will be able to store huge amounts of data where this can be accessed by several users at the same time. This allows healthcare professionals to receive real-time patient data as well as having the ability to manage and update these details when needed. These patient details could include information such as contact details and medical history. In addition, staff would be able to use this system to send vaccination reminders to patients who are registered online. This would save time and resources.

Although some healthcare practices have a computerized system, they do not offer online services for patients. Patients should have the convenience to access local GP services online from anywhere without the need to travel to the practice. Therefore, a solution for this problem is to create the system where patients would also have access to these services online.

A healthcare management system will be created for this project which can be accessed online and will be based on a database. Hence, it will be available to use on a range of devices with an internet connection. There will be several users however each user type will have a specific access to the system. This would allow them to view and manage information that is suited for them only. This project is important as there would be an electronic system available for a GP practice to use. This would benefit them as this system can be also used for multiple GP surgeries if required. Importantly, it would offer online services for patients to access anywhere at any time.

1.3 Aims and Objectives

1.3.1 Aim

To create a system that allows administrative staff and clinicians to work quickly and efficiently and to improve how patients approach their care.

1.3.2 Objectives

- Create a database which should be relevant to the healthcare management system and should be fully functional.
- Develop an appointment system that can support different types of appointments.
- Create a function that enables receptionists to send notifications to patients as a reminder for any vaccinations that are due soon.
- Develop a web-based system that allows users to store and manage data.

The project will deliver an online healthcare management system which can be only accessed by registered users. It will include an appointment service which enables users to book and manage appointments. In addition, there will be a user-friendly website which allows users to store and manage data such as patient details. They will also have access to other features on the system such as sending vaccination reminders and managing online prescriptions.

Chapter 2 – Literature Review

2.1 Introduction

This literature review will discuss about how personal information is processed lawfully in the UK and will explore the different areas of data protection. The system that will be developed for this project will mainly consists of sensitive data such as medical details of several patients. Therefore, it is important to have awareness of how this information should be handled and about the different security measures that could be taken to protect this data.

2.2. Data Protection Act

The Data Protection Act 1998 is a set of principles that businesses, organisations and the government have to follow to collect and use personal data for their own purposes. It covers information that is stored electronically and on paper. Importantly, it gives individuals their rights of access to personal data held by organisations about themselves. This act only regulates 'the processing of information relating to identifiable living individuals' (Room, 2007). Therefore, information that is related to companies, public authorities or charities will not be processed by this act as those are not considered as personal data.

The key activity that the act regulates is the 'processing' of personal data. The definition of processing can be considered as broad in relation to personal data. It can be interpreted as carrying out operations such as recording or holding this type of information. Other operations that are included can lead to disclosure or even destruction of this data (ICO, 2018). The purpose and the manner of processing of this personal data is determined by data controllers. Data controllers can be either organisations or individuals that are given responsibilities for data protection.

Any organisation that stores and handles personal information is required to register with the Information Commissioner (BBC 2018). It acts as an official body to enforce the legislation of the Data Protection Act 1998. Therefore, data controllers must inform them about the information that will be stored and how it will be processed in advance. These specific details will then be stored in the register which is under the control of the Information Commissioner's Office. The failure to notify this information will result in a criminal offence.

2.2.1 Data Protection Principles

Roebuck (2018) stated that 'the law balances the legitimate needs of the organisations to use personal data for businesses and the individual's right to respect for the privacy of their personal details.' This can be referred to the eight principles that is needed to be observed by data controllers when handling personal data. These are known as the Data Protection Principles.

1. The data should be fairly and lawfully processed.
2. The data should be obtained for specified and lawful purposes.
3. The data should be adequate, relevant and not excessive.
4. The data should be accurate and kept up to date.
5. The data should not be kept longer than necessary.
6. The data should be processed with an individual's rights.
7. Appropriate technical and organisational measures should be taken against unauthorised or unlawful processing of data and against accidental loss, destruction and damage.
8. The data should not be transferred to others countries outside the European Economic Area without adequate protection.

The first principle is about processing data fairly and lawfully. Therefore, the organisation should be fully transparent by informing the individuals they collected the data on and to let them know about how they intended to use that data for their purposes. They should also ensure that this done according the rules of the law. The second principle implies that the data that is obtained should be lawful and transparent. It should only be processed for the intended purpose hence it should not be processed in any other ways (Walker and Millman, 2018).

The third principle indicates that the data that was collected should be relevant and adequate for the original purpose. Excessive data should be ignored. The fourth principle suggests that organisation should review the personal data that is collected regularly to ensure that the information is accurate and up to date.

The fifth principle is about discarding information that is not longer needed for the purpose. It is advisable for organisations to have a review process. This will help them to take decisions on the time that is needed for retaining the information that is held and for discarding any irrelevant data. The sixth principle informs us that an individual has certain rights to their personal data that is held by the organisations. Some of these rights include changing inaccurate data and claiming compensation for damages caused by the breach of the act.

The seventh principle is about ensuring that the data that is kept should be secure. Therefore, it is important for organisations to be clear about the person who is responsible for the protection of data. This certain individual will ensure that certain security measures are in place so the organisation can be ready to respond to any breaches quickly and efficiently. The final eighth principle informs that any personal data should not be sent to countries outside the European

Economic Area unless the particular country ensures adequate data protection in relation to data processing (Walker and Millman, 2018).

2.2.2 Sensitive Personal Data

The Data Protection Act also provides protection for sensitive information and this can be defined as information that is in relation to racial or ethnic background, political opinions, religious beliefs, health conditions and criminal charges (GOV, 2018). However, further conditions are needed for data controllers to process this type of data. This is due to the sensitivity of the information as it could be used in a discriminatory way therefore it should be handled with greater care than personal data. One of these conditions can be that the individual has given explicit consent for the processing of their sensitive personal data. Another condition to process this data is that the processing is necessary for the purposes of complying with the employment law. In addition, it can also be necessary for legal proceedings such as establishing, exercising or defending the legal rights. These can be considered as some of the main conditions that organisations found as appropriate for processing sensitive information legally.

2.2.3 The Rights of Individuals

ICO (2018) has mentioned that 'the Data Protection Act gives right to individuals in respect of the personal data that the organisations hold about them.' These rights relate to the sixth data protection principle where it states that the data should be processed with an individual's rights.

- A right of access to a copy of the information comprised in their personal data.
- A right to object to processing that is likely to cause or is causing damage or distress.
- A right to prevent processing for direct marketing.
- A right to object to decisions being taken by automated means.
- A right in certain circumstances to have inaccurate personal data rectified, blocked, erased or destroyed.
- A right to claim compensation for damages caused by a breach of the Act.

A certain individual is entitled to access a copy of the information that is being held, along with a description of the particular information and the reasons for processing this data. However, the subject access request must be in writing in order for a data controller to verify their identity. The organisations are allowed to charge a fee up to £10 and they must respond to this request within 40 days (Jay and Clarke, 2010).

In addition, it gives an individual to prevent organisations from the use of the information that is being held about them if it causes unwarranted or substantial distress. They are also entitled to object their data from being processed for the purposes of direct marketing. Individuals can register with the Mailing Preference Service (MPS) to prevent unwanted mails being sent to them.

Furthermore, they have the rights for accessing information about the decisions that were made by automatic means. Decisions may be made due to the purpose of evaluation such as credit scores. They could request organisations to take their decisions by non-automated means. If inaccurate information about the individual is held by the organisation, the certain individual can order them to correct the inaccuracies. They also have the right to apply to court to deal with this issue. Individuals are entitled to compensation for any damages that were caused because the particular organisation

have breached a requirement of the Data Protection Act. However, evidence of this damage has to be shown to the court by the individual such as physical damages or tangible financial losses (Jay and Clarke, 2010).

These rights can be considered as significant and important to the Data Protection Act.

2.3 Data Security

Data has become as a critical asset in many organisations and can be considered as the key to growth and success. Kumar, Srivastava and Lazarevic (2005) states 'as the demand for data and information management increases, there is also a critical need for maintaining the security of database, applications and information systems.' This implies that there is an increase in the number of users who need instant access to sensitive or confidential information. Therefore, protective digital privacy measures have to be applied to prevent unauthorized access to databases, computers or websites. This is known as data security and is an essential aspect of IT organisations.

2.3.1 Threats to Databases

Databases can be considered as one of the most compromised assets as they are subject to a wide range of attacks. This is due to the importance of data for organisations as they require databases to store and manage information. However, this information can also include confidential details about the customers or about the organisation itself. When hackers gain access to this information, they are able to inflict damage and interrupt business operations. This may also cause a negative impact on the reputation of the organisation. Some of the most common threats to databases will be discussed in this section.

One of the most common attacks is caused by excessive privileges. When users are granted database privileges that exceeds their job functions, these may be used to gain access to confidential information. They may also use these privileges for unauthorized purposes by taking advantage of the vulnerabilities in the database system. SQL injection attacks can be considered as another common threat. This threat involves a user taking advantage of the vulnerabilities in the front end of web applications. Attackers can send unauthorized database queries which can give them escalated privileges. This may also give them unrestricted access to the entire database (Schulman, 2018).

Denial of Service attacks happens mostly through buffer flows, data corruption and consumption of resources. These types of attacks may cause disruption to the services on the system and can even cause the system to crash. Therefore, it will prevent users to have access to the database for the duration of the attack. Weak authentication is another common threat to databases as it allows attackers to steal the identity of a legitimate user to gain access to confidential information. It also allows it to be exposed to cryptanalytic attacks such as brute force attacks. This involves an attacker using a trial and error method to obtain information such as passwords. Moreover, data backups are often unprotected from attacks and this results in theft of data backups disks and tapes (Schulman, 2018).

However, it is possible to reduce the likelihood of these attacks happening with the use of different controls to properly protect the databases.

2.3.2 Data Protection Methods

There are several technologies that are available to protect data from unauthorized access such as authentication and encryption. It is also important that confidentiality, integrity and availability should be taken into consideration when protecting data, as they are considered as the foundation of data security.

Authentication is thought to be 'the outermost layer of security protection measures provided by the system' (Huang, 2013). The security of authentication can vary from password to biometric authentications. This method will verify an individual's identity before they are given access to the system and its data.

Encryption can be considered as one of the efficient methods of data protection. It is where digital data or hard drives are encrypted to prevent access from unauthorised users. It also ensures authenticity and integrity of the particular data.

In addition, data may no longer be accessible in a system due to damage or corruption. However, regular data backups can significantly ease the process of data recovery. Therefore, data should be backed up regularly using a physical data storage device or a server.

Firewalls acts as a barrier between internal and external networks. It can restrict access to all suspicious incoming and outgoing traffic. In addition, it can prevent malware infections that can cause corruption or damage to data.

These methods will ensure that data is protected from unauthorized users or malicious programs. However, data backups should also be carried out regularly to prevent loss of data.

2.3 Conclusion

This chapter has given a better insight into the different methods of data protection. From this literature review, it can be seen that data could be protected through technologies and by law. The Data Protection Act protects personal information and has given certain rights to the individuals whose data is held by organisations. Databases are also considered to be great importance to most organisations as they need those to store and manage data. However, the popularity of databases leads to several threats that may have impact on confidential information. These risks could be mitigated by using various methods to protect data.

These findings will be useful when implementing the system as importance will made on the security of the system. It is crucial for data protection as this system deals with patient medical details which can considered as confidential and sensitive. Therefore, various techniques will be implemented to prevent unauthorised access to the system and its data.

Requirements Analysis – Chapter 3

3.1 Introduction

Requirements analysis helps to determine the needs and conditions that have to be met for this project. It is considered as an important aspect of project management as it is significant to the overall success of the project. Various techniques will be used in order to derive the requirements for this system. Stakeholders will be identified and their benefits to the system will be explored to capture these requirements. Different types of use cases will be needed to understand the overall functionality of the system. In addition, this technique will help to gather functional and non-functional requirements. These will then be prioritised using MoSCoW analysis to understand the importance on the delivery of each requirement.

3.2 SWOT Analysis

SWOT analysis can be used to analyse this project strengths and weaknesses, opportunities and threats. It helps you to concentrate on the strengths and discover the opportunities that are available to you to improve this project. Also, it helps to identify the weaknesses and threats for this project earlier on so there will be more time this to solve these issues.

	Positive	Negative
Internal	<u>Strengths</u> <ul style="list-style-type: none">• Users can access the system at the same time on any device.• Patients can access local GP services online.• Access to several resources• Previous experience using similar technologies.• Users are able to store and manage data easily.	<u>Weaknesses</u> <ul style="list-style-type: none">• Lack of time to complete project.• Lack of experience and knowledge in some areas.• Pressure to meet deadlines.
External	<u>Opportunities</u> <ul style="list-style-type: none">• This system could be used as basis for a potential mobile app.• More functions can be added to improve the usability of the system.	<u>Threats</u> <ul style="list-style-type: none">• System may be vulnerable to malware and viruses.• Change in technology can cause the system to not work properly.• Maintenance from internet service providers could create problems.

There are many strengths and opportunities for this project which is shown in the table above. Therefore, it is ideal to proceed with this project. There are some weaknesses and threats for this project however these problems could be solved quite easily.

3.3 Stakeholder Analysis

A stakeholder is anyone who has an interest in business. Internal and external stakeholders can influence the activities of a business. There are also stakeholders for this project which are listed below.

Doctors & Nurses

They have patient consultations where they have responsibilities such as diagnosing an illness or giving vaccinations. This information will be recorded which will keep patient medical history up to date. Doctors also give prescriptions to patients if necessary.

Benefits:

The GP practice may not have an electronic system in place. Therefore, they will be able to benefit from using this Healthcare Management System as they will be able to store all their patient records in the system. This data can be easily changed or updated by them. In addition, they will be able to process any online repeat prescriptions that were requested from their patients.

Receptionists

Their role mainly involves with dealing with patients and administrative tasks such as booking appointments and filing.

Benefits:

The benefit will be that receptionists will be able to use the appointment service in this system. This appointment service would also be available to patients online but it will be limited. However, this will reduce the number of telephone calls to reception and save time for busy reception staff.

Patients

They book appointments at their GP surgery for medical reasons. They also any receive vaccination letters as a reminder for vaccinations that are due for them soon.

Benefits:

Patients can benefit from this system as they would have access to online services. These online services include booking appointments, viewing recent medical summary and requesting repeat prescriptions. They would also receive any notifications from the system if they have any vaccinations that are due soon. This would be convenient for patients as they do not have to wait for GP surgery opening times and can access these services at any time.

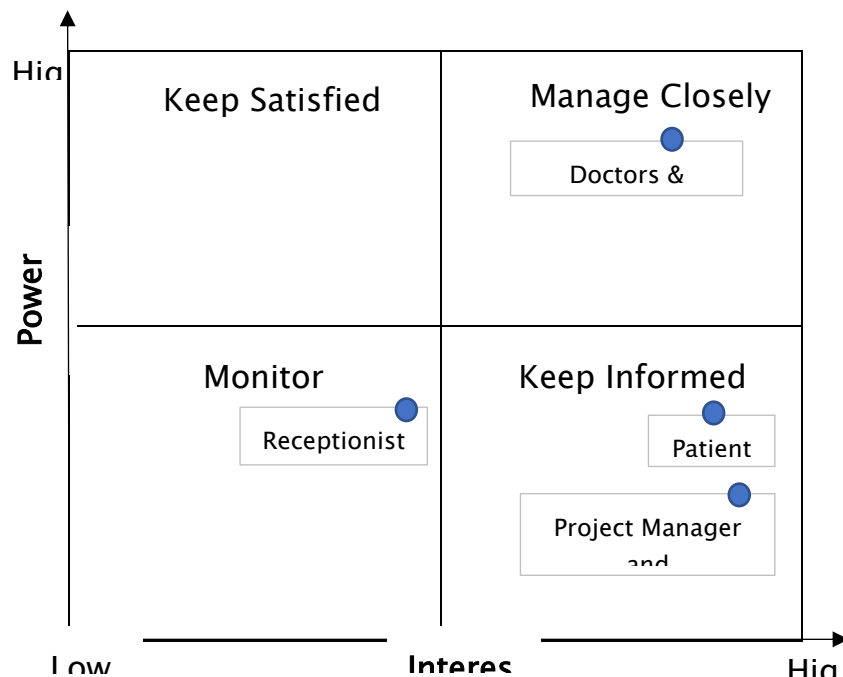
Project Manager and Software Developer

Project manager and software developer will be the same individual who will develop the Healthcare Management System. They will also have an important responsibility which is to manage the project effectively within a specific time frame.

Benefits:

This project will benefit them as it will help them to develop and improve their skills when creating the system.

3.4 Power/Interest Grid



The stakeholders that were identified earlier on are categorized based on their power and interest in the project. This Power/Interest grid shows the different ways on how to deal with these stakeholders. However, this grid has to be updated on regular basis to keep informed of any changes in the interest of these stakeholders during the project lifecycle. From this grid, doctors and nurses should be managed closely as they have high power and high interest in this project. They have a significant impact on this project therefore project decisions should be made with them carefully. Both receptionists and patients have low power but they should be informed and monitored. Communication with these stakeholders would be helpful to ensure that there are no major issues arising in this project.

3.5 Use Case Diagram

This diagram is a representation of the user's interactions with the healthcare management system by showing the relationship between the actors and the use cases. They help to visualize the functional requirements for each user of the system.

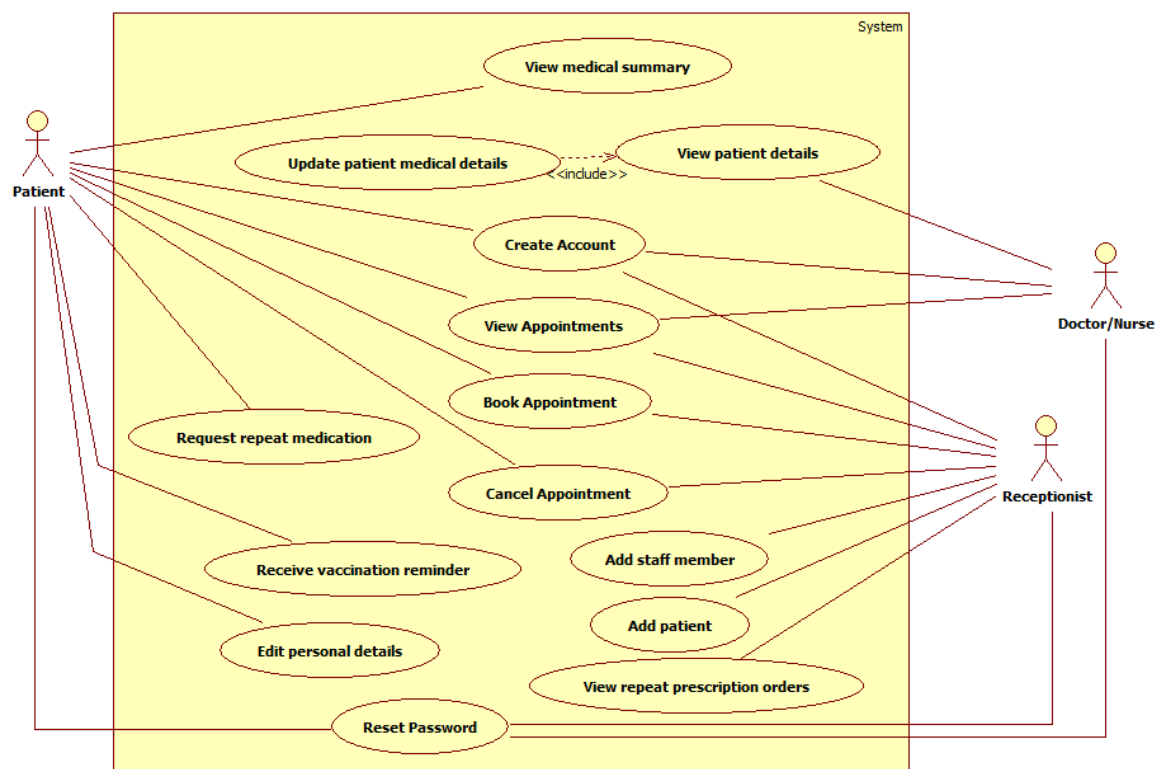


Figure 3.1 - Use Case Diagram

3.6 User Stories

User stories are brief description of a requirement from an end user perspective. The user story states the users and their possible actions on the system. These are written to influence the functionality of the system that will developed later on.

Receptionist

User Story ID	As a	I want to...	So that...
1.1	Receptionist	Add users	I can allow users to access the system.
1.2	Receptionist	Disable user's access	I can deny a user's access to the system if it is necessary.
1.3	Receptionist	Book appointment	I can arrange a consultation time between the patient and the required clinician.
1.4	Receptionist	Edit appointment	I can update the reason for the appointment if it is required by the patient.
1.5	Receptionist	Cancel appointment	I can cancel an appointment if it is no longer needed by the patient or clinician.

1.6	Receptionist	View appointment schedule	I can book appointments depending on the clinician's availability.
1.7	Receptionist	Create appointment slots	I can create available appointment times.
1.8	Receptionist	Delete appointment slots	I can remove appointment slots if required.

Doctor/Nurse

User Story ID	As a	I want to...	So that...
2.1	Doctor/Nurse	View appointments	I can view my schedule.
2.2	Doctor/Nurse	Add patient details	I can update patient details.
2.3	Doctor/Nurse	View patient details	I can view a patient's medical history.
2.4	Doctor/Nurse	Edit patient details	I can make changes to the patient details if it is incorrect.
2.5	Doctor	Create prescriptions	I can prescribe medications to my patients if needed.
2.6	Doctor	View prescriptions	I can check a patient's medication history.
2.7	Doctor	Approve repeat prescription requests	I can allow patients to receive their required prescriptions.
2.8	Doctor/Nurse	Add new vaccinations	I can update the vaccination list.
2.9	Doctor/Nurse	Edit vaccination list	I can edit details of vaccinations.

Patient

User Story ID	As a	I want to...	So that...
2.1	Patient	Update my contact details	I am able to let the GP surgery know about my updated details.
2.2	Patient	View appointments	I can view available and my booked appointments.
2.3	Patient	Book appointments	I can book appointments with a specific doctor.
2.4	Patient	Search appointments	I can find appointments that are suitable for the date and time chosen.
2.5	Patient	Cancel appointments	I can cancel an appointment if I no longer need it.
2.6	Patient	Update appointments	I can update the reason my appointment.
2.7	Patient	View prescriptions	I can view the medications that are available for me to request and my recent prescriptions.
2.8	Patient	Request repeat prescription	I can receive my medication if approved.

3.7 Use Case Texts

Use case texts show how a user will interact with the system to achieve a specific goal. Each use case text is represented as a sequence of steps that the user follows to complete that goal. They help to explain the system behaviour as it responds to the user's action and captures any potential errors that may occur during the process. These are considered as alternatives in these use case texts below.

Use Case	Book Appointment
Goal in Context	User of system books appointment
Preconditions	The user is logged in to the system.
Post conditions	Appointment slot is booked by the user.
Actors	Receptionist, Patient
Main Flow	<ol style="list-style-type: none">1. User clicks on the 'Appointments' tab in the homepage then clicks on 'Book Appointment' option.2. The system displays available appointments.3. User selects the required appointment and clicks on 'Book' option.4. The system displays a confirmation message.5. User clicks on 'Confirm' to book the appointment.6. The system displays a message telling the user that the appointment is booked successfully.
Alternatives	6a. If the appointment booking fails, the system shows an error message and asks the user to try again.

Use Case	View Appointments
Goal in Context	User can see a list of appointments that are specific to them.
Preconditions	The user is logged in to the system.
Post conditions	A list of appointments is displayed in the system.
Actors	Receptionist, Doctor/Nurse
Main Flow	<ol style="list-style-type: none">1. User clicks on the 'Appointment' tab in the homepage.2. The system displays a list of appointments that are specific to them.
Alternatives	N/A

Use Case	Update patient's medical details
Goal in Context	User can update the medical details of a patient.
Preconditions	User is on the required Patient page.
Post conditions	The patient's medical details are updated on the system.
Actors	Doctor/Nurse
Main Flow	<ol style="list-style-type: none">1. User clicks on the 'Add Clinical Data' tab on the Patient page.

	<ol style="list-style-type: none"> The system displays a relevant data entry form. User completes the form as required. User clicks on 'Save' option. The system displays a success message and updates the database.
Alternatives	<p>5a. An error message would be shown if the database cannot be updated.</p> <p>5b. Validation error messages would be shown if the details that are entered do not support the validation rules.</p>

Use Case	Request Repeat Medication
Goal in Context	User of system requests repeat medication.
Preconditions	The user is logged in to the system.
Post conditions	The prescription for the medication is requested by the user.
Actors	Patient
Main Flow	<ol style="list-style-type: none"> User clicks on the 'Prescriptions' tab in the homepage and then clicks on 'Request Repeat Prescription' option. The system displays available repeat prescriptions for that user. User selects the required repeat prescription(s) and clicks on 'Request' option. The system displays a confirmation message. User clicks on 'Confirm' to confirm their prescription requests. The system displays a success message telling the user that their request has been sent to their practice.
Alternatives	6a. If the prescription request fails, the system shows an error message and asks the user to try again.

Use Case	Add Staff Member
Goal in Context	User can add details of a new staff member onto the system.
Preconditions	The user is logged in to the system.
Post conditions	The new user's details are successfully added to the system.
Actors	Receptionist
Main Flow	<ol style="list-style-type: none"> User clicks on the 'Users' tab and then clicks on 'Add New User' option. The system displays a relevant data entry form. User completes the form as required. User clicks on 'Add' option. The system displays a success message

	and updates the database.
Alternatives	<p>5a. An error message would be shown if the database cannot be updated.</p> <p>5b. Validation error messages would be shown if the details that are entered do not support the validation rules.</p>

3.8 Functional and Non-Functional Requirements with MoSCoW

Functional Requirements

Key

Receptionist – R

Doctor/Nurse – D/N

Patient – P

FUNCTIONALITY	REQ ID	REQUIREMENT	PRIORITY	USER
Manage Users	1.1	Add users	MUST	Receptionist
	1.2	View users	SHOULD	Receptionist
	1.3	Disable user's access	COULD	Receptionist
Manage Patient Details	2.1	Add patient details	MUST	D/N, R
	2.2	View patient details	MUST	D/N, P
	2.3	Edit patient details	MUST	D/N, R, P
Manage Availability	3.1	Create appointment slots	MUST	Receptionist
	3.2	View appointment schedule	MUST	Receptionist
	3.3	Delete appointment slots	MUST	Receptionist
Manage Appointments	4.1	View appointments	MUST	R, D/N, P
	4.2	Book appointments	MUST	Receptionist, Patient
	4.3	Search appointments	COULD	Receptionist, Patient
	4.4	Cancel appointments	MUST	Patient, Receptionist
	4.5	Update appointments	MUST	Receptionist, Patient
Manage Prescriptions	5.1	Create prescriptions	MUST	Doctor
	5.2	View prescriptions	MUST	Patient, Doctor
	5.3	Request repeat prescriptions	SHOULD	Patient
	5.4	Approve repeat prescription requests	SHOULD	Doctor
Manage Vaccinations	6.1	Add new vaccinations	MUST	D/N, R
	6.2	Edit vaccination list	MUST	D/N, R
	6.3	Send vaccination notifications	SHOULD	Receptionist
Manage Communication	7.1	Send instant messages	WON'T	P, D/N, R

3.9 Non – Functional Requirements

TYPE	REQUIREMENT	PRIORITY
------	-------------	----------

Accessibility	Users should be able to access the system anywhere on any device therefore it should be responsive.	MUST
Accessibility	Several users should be able to access the system at the same time.	MUST
Availability	The system should be available to use 24/7 to all users who have internet connection.	MUST
Performance	The system should be able to handle large amounts of data.	MUST
Performance	The system should be able to respond to the user's actions instantly.	MUST
Accuracy	Users should be able to view real-time data therefore the system should be able to update data as required and immediately.	MUST
Reliability	The system should be reliable so it should not crash while in use.	MUST
Usability	The system should be user-friendly therefore it should be easy to use and understand by all users.	MUST
Security	Only registered users can have access to the system.	MUST
Security	Passwords should be encrypted to prevent unauthorized access the system.	MUST

Chapter 4 - Design

4.1 Introduction

The design chapter will focus on producing suitable user interface and database designs for this system. It is an essential phase of the software development cycle therefore careful design decisions will be made as poor designs leads to complications during the implementation stage. Several known design principles and processes will be applied in order to develop high quality designs for this web application. These include activity diagrams, entity-relationship diagram, wireframes and mock ups. The requirements that were found during requirements analysis will also be taken into consideration when designing this system. This is important as it will help in developing designs for the required functionalities as well as designing a system that meets users' expectations.

4.2 Database Design

The importance of database design can be considered as critical to this project. This is due to the significance of data to this system and its users. A well-designed database ensures consistent data and better performance. Hence, this database should be designed to only store useful and required data which can be managed by different users on this system. Normalization technique will also be used for designing this database to improve the database structure and reduce data redundancy.

4.2.1 Entity Relationship Diagram

An entity relationship diagram is a data modelling technique and a graphical representation of the entities and their relationships to each other. It can be considered that this diagram illustrates the logical structure of a database. It also consists of attributes which can be categorized using different keys such as primary and foreign keys. Primary keys uniquely identify an entity while foreign keys establish relationships between tables in the most efficient manner. The relationships between

entities can be further defined by cardinalities by placing relationships in context of numbers. Each of them is represented by the number of instances of an entity that can be associated with each instance of another entity.

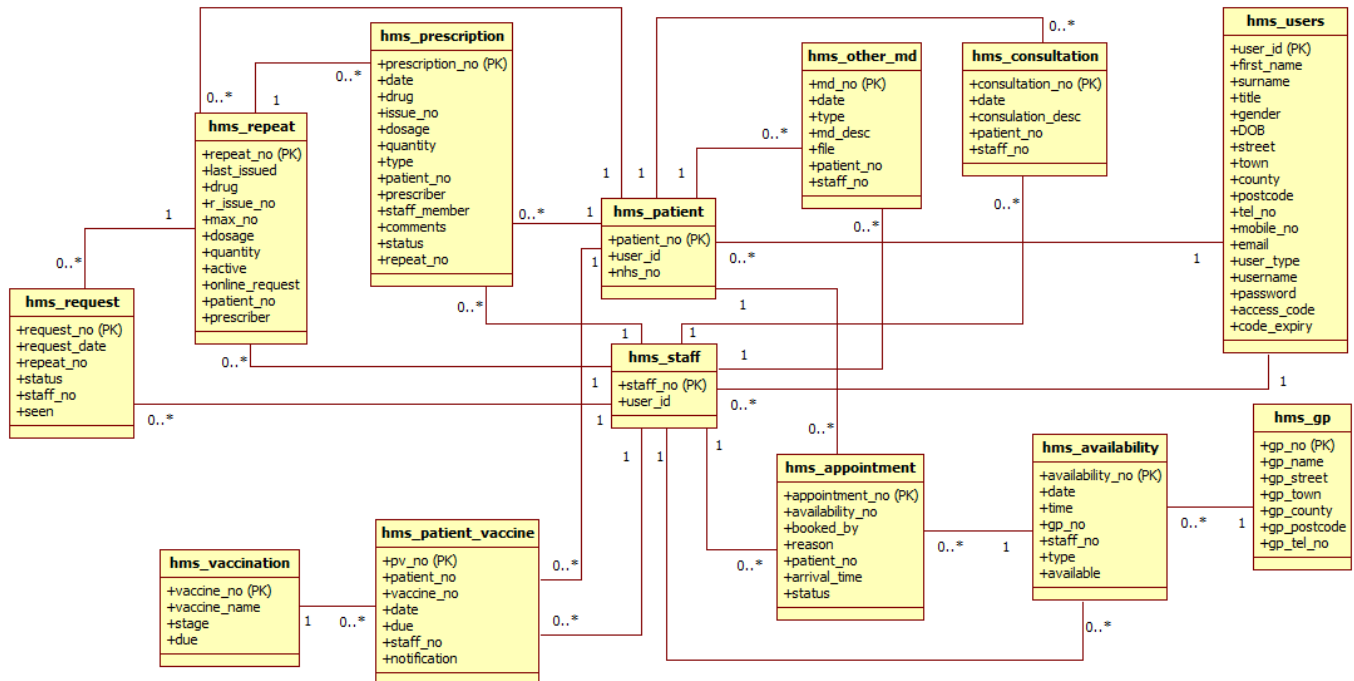


Figure 4.1 - Entity Relationship Diagram

The entity relationship diagram shown on Figure 4.1 is a revised version of the previous initial ER diagram that was designed at an earlier stage of this project. Several changes were made in order to improve the database structure and to support additional functionalities of the system. This was achieved by introducing additional tables and changing some of the attributes. The database was also normalised to third normal form by forming new relationships between these tables. This helps to manipulate data quickly and efficiently without compromising data integrity. Furthermore, this will also improve the performance of this system as data can be retrieved quickly through the use of efficient joins.

Five main areas can be seen in this ER diagram which illustrates some of the main functionalities of the system. These areas include 'Users', 'Appointments', 'Prescriptions', 'Vaccinations' and 'Consultations & Other Medical Details'. Users section involves with recording contact information

and login details for each user. In addition, users are classified into different user types such as patients and staff. This avoids duplication of data as common information of both users will be recorded onto one table called 'Users'. Appointments section handles with information about the availability of clinicians and certain information about each appointment. Prescriptions section deals with different types of prescriptions such as acute and repeat prescriptions for each patient. Requests for repeat medication will also be stored in the Requests table. Furthermore, vaccinations section will contain information about different vaccinations and the vaccines that may be given to patients. Consultations and other medical details section will also store information about each consultation and medical details of each patient.

4.2.2 Assumptions

Assumptions have been made from analysing the ER diagram in order to have a better understanding of how the system should handle certain data and how the database should be developed.

- Initially, all users will not have login details when they added to the database. This information would be updated once the users register to access the services on the system.
- Repeat medication requests can only be made if certain repeat prescriptions are available to a patient. These repeat prescriptions would be initially created by clinicians.
- All repeat medication will be recorded as active when added to the database. This can be updated by a clinician if necessary.
- Appointments can only be created if the required availability slots are available.
- Patients can be given many vaccines.

4.2.3 Data Dictionary

A data dictionary is a set of information which describes the structure and format of data that is collected within the database. It also includes information such as field length, constraints and default values. This would be helpful in the development and maintenance of the database for this system.

Relation Name	Attribute Name	Data Type	Length	PK/FK	Not Null	Other Constraints
hms_users	user_id	INT	7	PK	✓	
	first_name	VARCHAR	30		✓	
	surname	VARCHAR	30		✓	
	title	VARCHAR	10		✓	
	gender	VARCHAR	10		✓	
	DOB	DATE			✓	
	street	VARCHAR	50		✓	
	town	VARCHAR	35		✓	
	county	VARCHAR	35		✓	
	postcode	VARCHAR	10		✓	
	tel_no	VARCHAR	11		✓	
	mobile_no	VARCHAR	11		✓	
	email	VARCHAR	50		✓	
	user_type	VARCHAR	20		✓	
	username	VARCHAR	50			
	password	VARCHAR	50			
	access_code	VARCHAR	20			

	code_expiry	DATETIME				
hms_patient	patient_no	INT	7	PK	✓	
	user_id	INT	7	FK	✓	
	nhs_no	BIGINT	10		✓	
hms_staff	staff_no	INT	7	PK	✓	
	user_id	INT	7	FK	✓	
hms_gp	gp_no	INT	7	PK	✓	
	gp_name	VARCHAR	30		✓	
	gp_street	VARCHAR	50		✓	
	gp_town	VARCHAR	30		✓	
	gp_county	VARCHAR	30		✓	
	gp_postcode	VARCHAR	10		✓	
	gp_tel_no	VARCHAR	11		✓	
hms_availability	availability_no	INT	7	PK	✓	
	date	DATE			✓	
	time	TIME			✓	
	gp_no	INT	7	FK	✓	
	staff_no	INT	7	FK	✓	
	type	VARCHAR	30		✓	
	available	VARCHAR	10		✓	
hms_appointment	appointment_no	INT	7	PK	✓	
	availability_no	INT	7	FK	✓	
	booked_by	INT	7	FK	✓	
	reason	VARCHAR	50		✓	
	patient_no	INT	7	PK	✓	
	arrival_time	TIME				
	status	VARCHAR	30		✓	Default '-----'
hms_consultation	consulation_no	INT	7	PK	✓	
	date	DATE			✓	
	consulation_desc	VARCHAR	255		✓	
	patient_no	INT	7	FK	✓	
	staff_no	INT	7	FK	✓	
hms_prescription	prescription_no	INT	7	PK	✓	
	date	DATE			✓	
	drug	VARCHAR	50		✓	
	issue_no	INT	5			
	dosage	VARCHAR	30		✓	
	quantity	INT	5		✓	
	type	VARCHAR	20		✓	
	patient_no	INT	7	FK	✓	
	prescriber	INT	7	FK	✓	
	staff_member	INT	7	FK	✓	
	comments	VARCHAR	100		✓	Default '-----'
	status	VARCHAR	30			Default 'Given'
	repeat_no	INT	7	FK		
hms_repeat	repeat_no	INT	7	PK	✓	
	last_issued	DATE				
	drug	VARCHAR	30		✓	
	r_issue_no	INT	11		✓	Default '0'
	max_no	INT	11		✓	

	dosage	VARCHAR	30		✓	
	quantity	INT	11		✓	
	active	VARCHAR	10		✓	Default 'Yes'
	online_request	VARCHAR	5		✓	
	patient_no	INT	7	FK	✓	
	prescriber	INT	7	FK	✓	
hms_request	request_no	INT	7	PK	✓	
	request_date	DATE			✓	
	repeat_no	INT	7	FK	✓	
	status	VARCHAR	30		✓	
	staff_no	INT	7	FK		
	seen	DATE				
hms_vaccination	vaccine_no	INT	7	PK	✓	
	vaccine_name	VARCHAR	30		✓	
	stage	VARCHAR	10			
	due	INT	5			
hms_patient_vaccine	pv_no	INT	7	PK	✓	
	patient_no	INT	7	FK	✓	
	vaccine_no	INT	7	FK	✓	
	date	DATE			✓	
	due	DATE				
	staff_no	INT	7	FK	✓	
	notification	VARCHAR	20			
hms_other_md	md_no	INT	7	PK	✓	
	date	DATE			✓	
	type	VARCHAR	30		✓	
	md_desc	VARCHAR	100		✓	
	file	VARCHAR	50			
	patient_no	INT	7	FK	✓	
	staff_no	INT	7	FK	✓	

Table 1 - Data Dictionary

All telephone and mobile numbers have a data type of VARCHAR as INT ignores the first zero in a set of numbers. The code_expiry column will record the expiry dates which will be generated based on the time that the access codes are given to the users for registration. DATETIME has been used for this attribute as it can store both date and time values which will improve the accuracy of the expiry dates. BIGINT has been used for NHS numbers as INT data type is not sufficient for storing higher values of 10-digit numbers. In addition, only file paths will be recorded on the file column in the 'hms_other_md' table therefore VARCHAR data type was used. This was because files require extensive storage space and this will lower the performance when retrieving data from the database. Therefore, it was decided that files should only be stored in the server.

Many of the constraints shown on Table 1 would be achieved by validating data on the web application before it is added to the database. This will avoid incorrect data being entered into the database and helps to improve the performance of the system.

4.3 User Interface Design

This section of the chapter will explore the different approaches that were taken to design this web application such as wireframes and mock ups. It will also include information about the structure and navigation of this website.

4.3.1 Website Structure and Navigation

The structure of the website is based on the functional requirements that were previously found during the requirements analysis. Every user type has a unique navigation on the system where they would be able to access particular web pages. However, all users will need to login to access them. The structure and navigation of this system will be shown on hierarchy diagrams which will be specific to each user type. These will consist of different pages and functionalities required for certain users. Importance was made on navigation to ensure that the website is structured and organised. It was also considered it should be consistent and clear to understand for all users. Below are the navigation diagrams for all three user types.

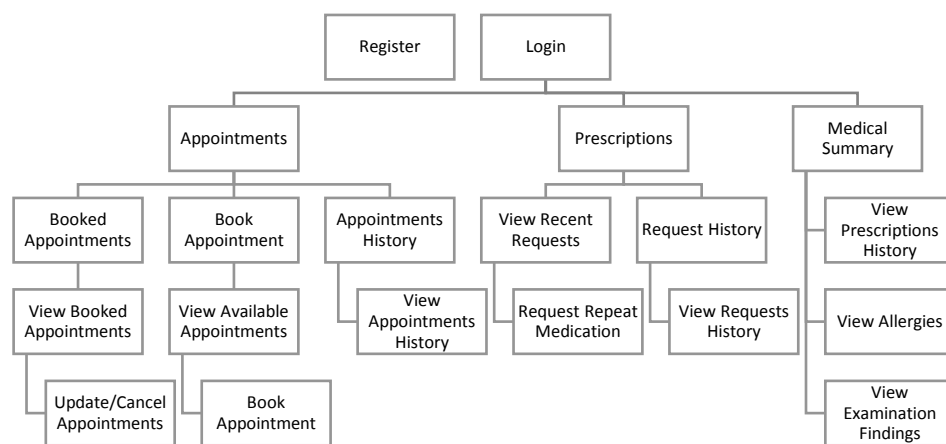


Figure 4.2 - Navigation Diagram for Patients

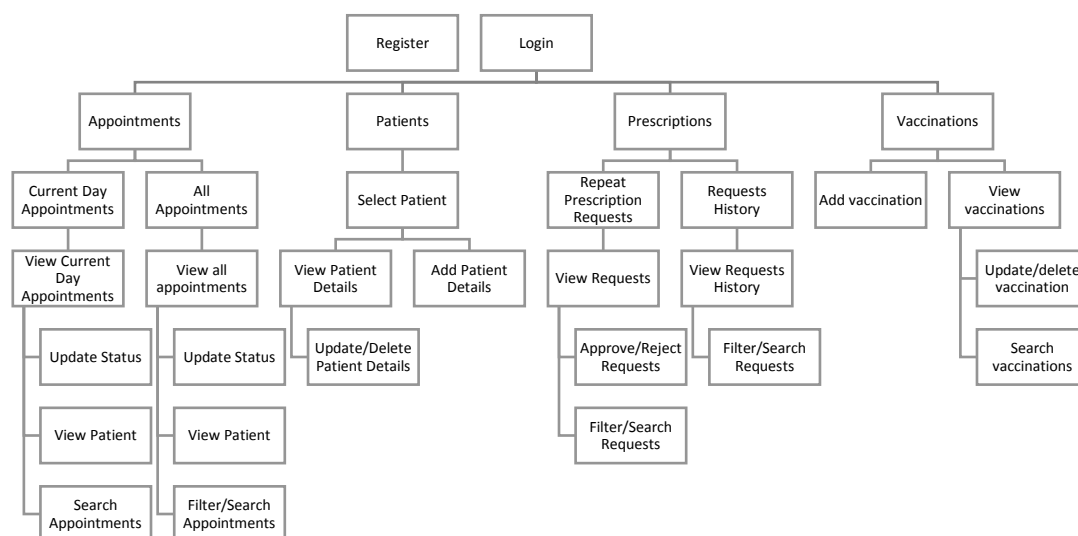


Figure 4.3 - Navigation Diagram for Doctors/Nurses

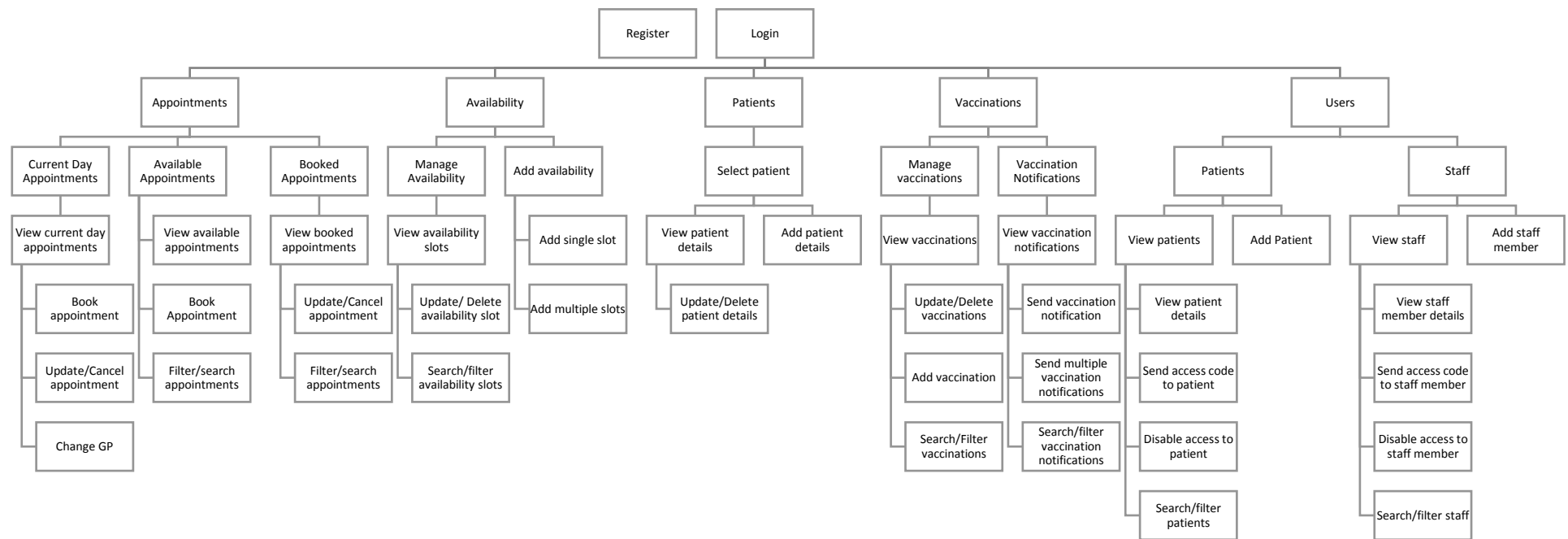


Figure 4.4 - Navigation Diagram for Receptionists

These diagrams illustrate that receptionists are associated with many different functionalities compared to other user types. This also shows that they have more responsibilities towards patients and clinicians.

However, similar functionalities can be seen between clinicians and receptionists. They both manage vaccinations which can be seen on Figure 4.3 and 4.4. This page will be shown to both users however clinicians are not required to send vaccination notifications. Therefore, the hyperlink on this page for vaccination notifications will be hidden from clinicians which will prevent them from accessing this particular page and its functionalities. This will also reduce development time as additional pages do not need to be created for this feature to work.

4.3.2 Activity Diagrams

An activity diagram is a visual representation of a workflow of an activity which describes a functionality that will be used in the system. It will also show any alternative paths through the workflows if required. These types of diagrams will generally show a series of actions that is needed to achieve a particular goal. This helps to understand the different processes and behaviour of a system.

Activity diagrams were developed to represent the workflows of some of the functionalities that will be needed for the system.

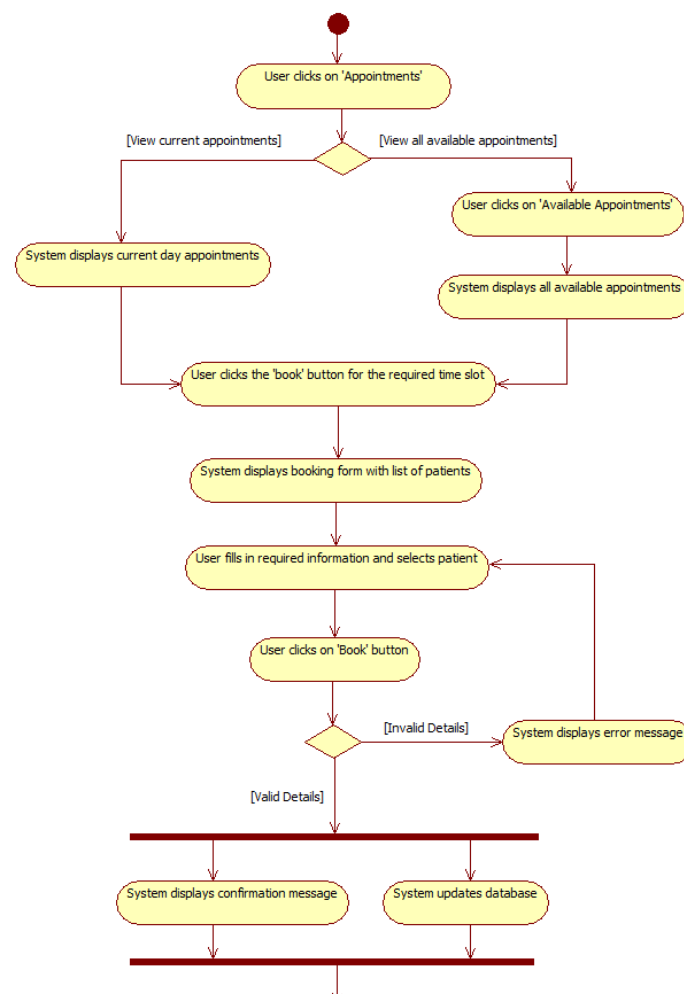


Figure 4.5 - Book Appointment Activity Diagram

Figure 4.5 shows an activity diagram that represents the workflow of booking appointments for receptionists. It illustrates that this goal can be achieved by two different ways. They could book an appointment by selecting one of the time slots in 'current day appointments'. Another way is to book an appointment by selecting a time slot that is shown in the list of all available appointments. However, both ways require valid details in order to book an appointment. A confirmation message will be then shown to the user if their booking is successful.

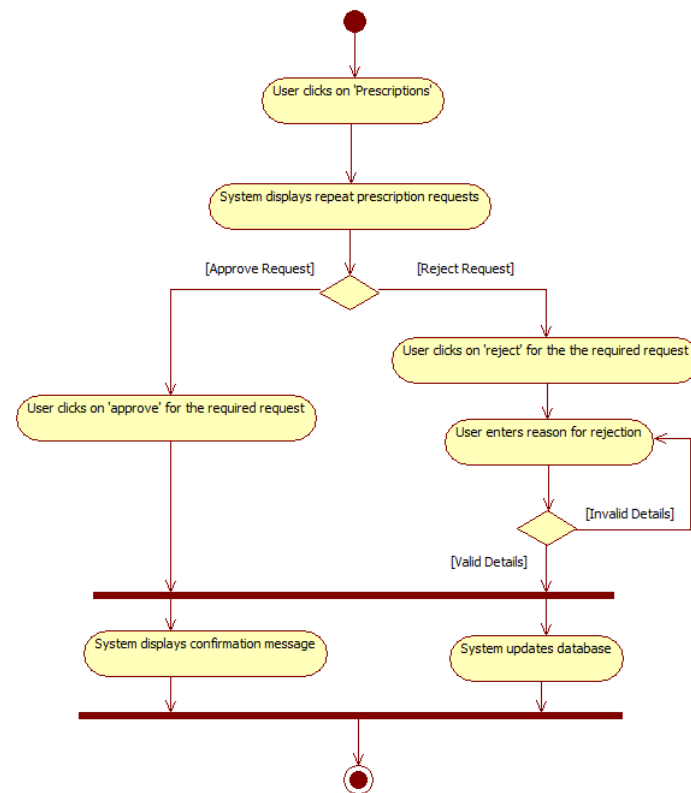


Figure 4.6 - Manage Repeat Prescription Request Activity Diagram

Figure 4.6 shows another activity diagram that represents the workflow of managing repeat prescription requests for clinicians. Clinicians will have a choice to either approve or reject this request. However, if they choose to reject this prescription request, they would need to enter a valid reason in order to proceed with this action. Concurrent activities can be also seen in this diagram where the system updates the database and displays a confirmation message to the user.

4.3.3 Sequence Diagram

Sequence diagrams are used to show the interactions between the objects in the order of how these interactions occur. It models the flow of logic of a functionality in a visual representation. This also helps to understand the users and objects that are involved and how they interact with each other to complete a process. These diagrams can be useful when designing a system as it represents the requirements of the system.

A sequence diagram has been designed to illustrate the interactions between different objects in a system for a particular functionality.

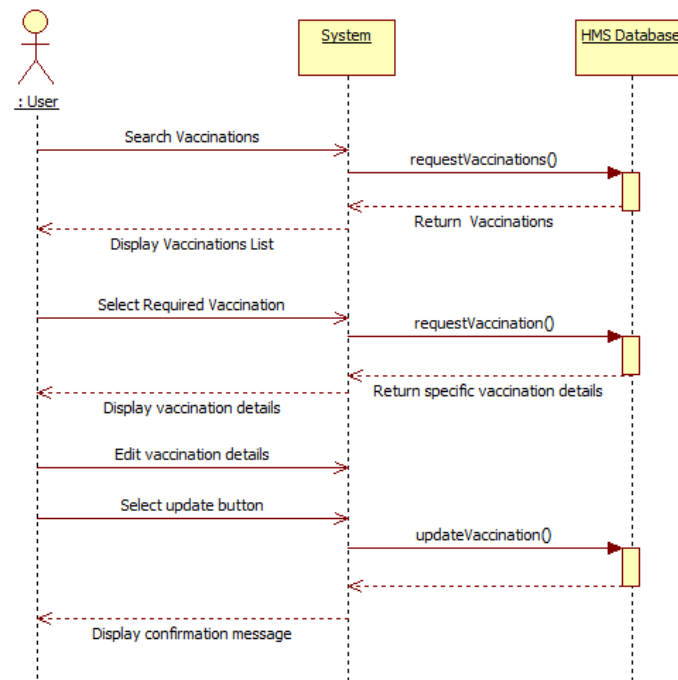


Figure 4.7 - Update Vaccination Sequence Diagram

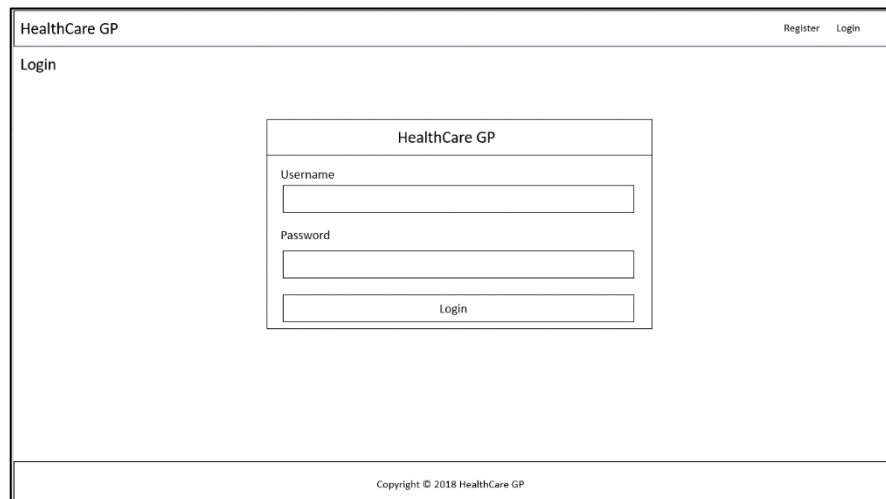
Figure 4.7 shows a sequence diagram that describes the process of updating a vaccination for clinicians. The first step is executed at the user interface level where the user searches for the vaccinations. The specific data is then returned as a list of vaccinations from the database. The user then selects the required vaccination and these details are also returned from the database. The user then edits the required fields and clicks on the update button. The system updates these details on the database and a confirmation message will be shown back to the user. This diagram can be considered as more detailed in comparison to the activity diagrams. The reason is that it shows the interactions beyond the user interface level as a database is also involved.

4.3.4 Sketches

Sketches are useful during the initial stages of designing as changes could be made easily by simply erasing the particular areas. It also helps to visualise the project quickly without the need of a software. These are generally used to express ideas and explore various design concepts. Sketches can be considered as low fidelity prototypes as only the key elements will be shown and there is no interactivity involved. Some sketches were drawn to visualise the navigation and page layouts of the system. These can be seen below in Figure 4.8.

4.3.5 Wireframes

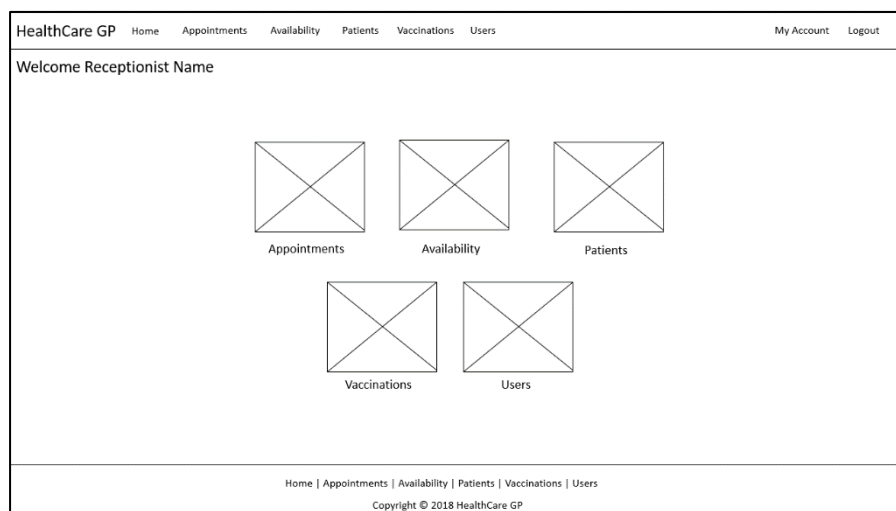
Wireframes are important as it allows you to plan the layout and the interactions of a system carefully without any distractions such as colours. It provides a basic structure of the system and it helps to identify any changes that need to be made without any difficulties. A number of wireframes were developed using Microsoft PowerPoint which are more detailed than the previous sketches. Technologies have been also taken into consideration when designing as this may be beneficial when developing the system. Additional wireframes can be seen in Appendix B.



The wireframe shows a login page for 'HealthCare GP'. At the top, there is a header bar with 'HealthCare GP' on the left and 'Register' and 'Login' links on the right. Below the header, the word 'Login' is displayed. The main content area features a central login form with a title 'HealthCare GP', followed by 'Username' and 'Password' labels, each with a corresponding input field, and a 'Login' button at the bottom. A footer bar at the very bottom contains the text 'Copyright © 2018 HealthCare GP'.

Figure 4.9 - Login Page

The login page is considered as the main page for all users which is shown on Figure 4.9. It consists of a single login form that asks the user for the username and password. This page is designed to be simple as it will bring focus on the login form. It was decided that only one login page should be designed for all user types. This feature would avoid confusion between different users and it will be easier for them to access certain webpages.



The wireframe shows a receptionist homepage for 'HealthCare GP'. The header bar includes 'HealthCare GP' and a navigation menu with 'Home', 'Appointments', 'Availability', 'Patients', 'Vaccinations', and 'Users'. On the right side of the header are 'My Account' and 'Logout' links. Below the header, a personalized greeting 'Welcome Receptionist Name' is shown. The main area contains five placeholder boxes, each with a diagonal cross, representing images for 'Appointments', 'Availability', 'Patients', 'Vaccinations', and 'Users'. A footer bar at the bottom displays a breadcrumb trail 'Home | Appointments | Availability | Patients | Vaccinations | Users' and the copyright notice 'Copyright © 2018 HealthCare GP'.

Figure 4.10 - Receptionist Homepage

Figure 4.10 represents the receptionist homepage which is the page they will see when they are logged in. This page consists of the main menu where they will be able to access other pages using the links on the navigation bar or using specific buttons. Icons will be used to represent these buttons as this helps to recognise the functionalities more easily. They are also considered to be more visually appealing to the users. The homepages for clinicians and patients will also be similar as it important to have a consistent layout throughout the website.

Figure 4.11 - Available Appointment Page

Another important page can be seen on Figure 4.11 which shows the available appointments page for receptionists. This page will show all the available appointments that can be booked for patients. This information will be displayed in a table as it will look more organised and presentable to the users. An important design decision was made to include a filter for this table. The default Bootstrap filters were not considered as ideal for filtering this data. This is because it does not provide the option for the user to choose and filter specific data. Therefore, this filter was designed to overcome this issue. Date pickers will also be used for selecting specific dates in this filter as it will avoid users entering incorrect format of dates.

4.3.6 Mock Ups

Mock ups can be considered as mid-to-high fidelity visual representations of a system which also offers a user-friendly interface. It will include interactive elements such as buttons and icons which represents certain functionalities. These mock ups can be used as a basis for designing the web application.

HealthCare GP Home Appointments Availability Patients Vaccinations Users My Account Logout

Patient Details

Patient: Hayden Conway Change Add

Personal Details Consultations Medical History Prescriptions Vaccinations Other

NHS Number: 4561237891

Title: Mr

First Name: Hayden

Surname: Conway

Date of Birth: 06/04/1954

Gender: Male

Address

House No/ Street: 27 Main Road

Town: Charlton

County: London

Postcode: SE2 7RF

Contact Details

Telephone No: 02083501521

Mobile No: 07237089384

Email: haycon@gmail.com

Update

Figure 4.12 - Patient Details Page

The Patient Details page shown on Figure 4.12 can be considered as one of the main pages of this system. This page was designed carefully it will deal with crucial data that are associated with several patients. It was decided that medical details should be separated into different tabs. The headings of these tabs will represent the information that it will contain such as consultations or prescriptions. Most of the tabs will contain information presented in a table format apart from the Personal Details tab. This tab will contain several input fields which are organised into different columns depending on the type of information. These design decisions were made to improve the accessibility of information for clinicians. They do not need to navigate to different pages for particular information as this design allows all the necessary information to be included onto a single page.

In addition, the functionalities for adding certain medical details for a patient will be presented inside a dropdown button. This will look more organised compared to designing several buttons for each of these functionalities. It was also decided that all of this buttons on this page will either show a modal form or the user when be useful for they will be the required the related all on one page.

a modal form or the user when be useful for they will be the required the related all on one page.

a message to clicked. This will the clinicians as able to access information and functionalities

Book Appointment

Appointment Details:

Type: General

Date: 20/04/2018 Time: 13:00

Clinician: Hannah

Reason:

Patient:

Search..

First Name	Surname	DOB	Address	Telephone No.
<input type="checkbox"/> Kelly	Rogers	10/07/1982	75 Somerset Drive, Greenwich, London, SE1 7SB	02086543983
<input type="checkbox"/> Harry	Mills	30/01/1999	351 Vanburgh Lane, Blackheath, London, SE6 2VJ	02085467438
<input type="checkbox"/> Brooke	Sullivan	23/12/1993	25 Southern Way, Charlton, London, SE22 7LX	02085647812

Figure 4.13 - Appointment Booking Modal Form

It is decided that modal forms will be one of the main design features that will be used in this system and an example can be seen on Figure 4.13. This shows an appointment booking form for receptionists. They will be required to enter a reason and select a patient to book an appointment. These types of modals will be shown when clicked on certain buttons on the webpages. A search functionality will also be included in this booking modal as it will be useful for receptionists to search for a particular patient. This was considered as an important design feature as it would be impractical to find a particular patient by just looking through a table of records. Specific details will also be shown on the modals to ensure users that they selected the correct data. Dynamic web pages can be created with the use of modals and this reduces the amount of static html pages that are needed. These modals can also be shown on the same page and this will save time for users as functionalities can be easily accessed.

HealthCare GP
Home
Appointments
Prescriptions
Medical Summary
My Account
Logout

Prescriptions

[Requests History](#)

Recent Requests

Request Date	Drug	Status
Data not Found		

Available Repeat Prescriptions

	Last Issued	Drug	Dosage	Quantity	Available Requests
<input type="checkbox"/>	24/02/2018	Simvastatin 10mg	Once a day	30	3

Request

Figure 4.14 - Repeat Prescriptions Page

This page will display the recent requests and the available repeat prescriptions for a particular patient as shown on Figure 4.14. This information will be separated into two separate tables. The recent requests table will only show requests for the past month. It will also include a status where it

will display the feedbacks that are given by the clinician. The available repeat prescriptions table will list the repeat prescriptions that are available for a particular patient. A checkbox will also be created for each row in this table as it will allow the patient to check the required prescriptions to request. This will save time for patients as they do not need to request for each prescription individually. This design decision was made in order to make the system more efficient for users.

The table's design and layout presented in Figure 4.14 will be the same for all other tables in this web application. This design choice was made to maintain consistency throughout the system. Also, blue colour schemes were seen in many healthcare websites based on the research that was undertaken for this project. Therefore, it was decided that a blue colour scheme should be used for this system as well, as it was considered as appropriate for all types of users.

4.4 Conclusion

This chapter can be considered as an essential aspect of the project as important decisions were made on designing the database and the user interface of the system. Data modelling techniques such as creating an entity relationship diagram were used. This helped to identify the different tables and relationships that are needed to develop the database. In addition, the website structure was then developed to demonstrate the navigation of the system. It shows the how the different pages in the system are linked together. It also shows the various routes that a user need to take to navigate to certain pages or perform certain actions. Activity and sequence diagrams were designed to represent the series of steps that are needed to achieve certain functionalities. These helped to think about how the website should be designed to support the database and workflows of these different activities.

Afterwards, several sketches, wireframes and mock ups were produced to give visual representations of the system. Mock ups will be used when developing the system as they were designed to be close representations of certain web pages and functionalities. They also contain many important design features that should be included in the system.

Designing the database was considered as challenging because it was important to ensure that the database design is correct. The database that will be developed plays a key role in this system as storing and managing data is essential for its users. However, extensive research and careful design decisions were made to overcome this issue.

Overall, these designs can be used as a basis for implementing the system as it helps to understand the layout and the navigation of the system.

Chapter 5 – Implementation

5.1 Introduction

This chapter will focus on the development of the system and will discuss about the different technologies and tools that were used to create this system. It will also include about the process of creating the database and the user interface for this system. In addition, there will be a discussion on the various decisions that were made including any challenges that arose during implementation.

5.2 Technologies and Architecture

5.2.1 Technologies

Dreamweaver

Dreamweaver was chosen as the text editor for implementing the system as it highlights any syntax errors. This was useful when writing code as errors can be identified instantly which saves time when developing the system. This development tool also includes features such as the ability to view visual designs of the particular webpages within the application.

Bootstrap

It is a front-end web framework that allows websites to be responsive. Bootstrap's CDN could be added to the site which can make it responsive easily. Designing the system using CSS requires extensive time to design the application therefore it was decided that Bootstrap is an ideal choice for designing and making the system responsive.

WinSCP

This is a FTP software which its main function is to transfer files from a local computer to a server with a secure connection. It also supports file synchronization as this is useful when making changes in code. This would be automatically reflected in the files that are stored in the web server. The university's own web server (KUNET Webserver) will be used and the database will be managed using PhpMyAdmin.

MySQL

This is a relational database management system where this will be used to create a database for the system.

jQuery

It is a JavaScript library which simplifies the scripting of HTML as it requires less lines of code to achieve a particular outcome than using JavaScript. This saves time during implementation.

AJAX

Better user experience can be achieved through the use of Ajax as the user does not need to refresh the page to see an updated content. It is able to send and retrieve data asynchronously without any interference of the existing page. This improves the overall usability and performance of a system.

5.2.2 Architecture

The system architecture defines the structure and behaviour of the system. The system will be created as a web application. It will also be accessible on any device ranging from computers to mobile phones. It will be hosted on a server which would allow users to access this application more easily.

5.3 Database Implementation

5.3.1 Creation of Tables

The tables were created using the ER diagram that was designed in the design phase. This helped in building the structure of each table. This database would be managed using PhpMyAdmin that is included in the server.

There are 13 tables in total and it would be time consuming if it was created using SQL statements. Therefore, it was created using the web application that is provided by PhpMyAdmin. The structure of the columns of each table can be configured easily using this approach. In addition, it ensures that tables can be created correctly. This application does not allow the creation of tables when incorrect data is entered. This will be displayed as an error message that will give clear descriptions of the

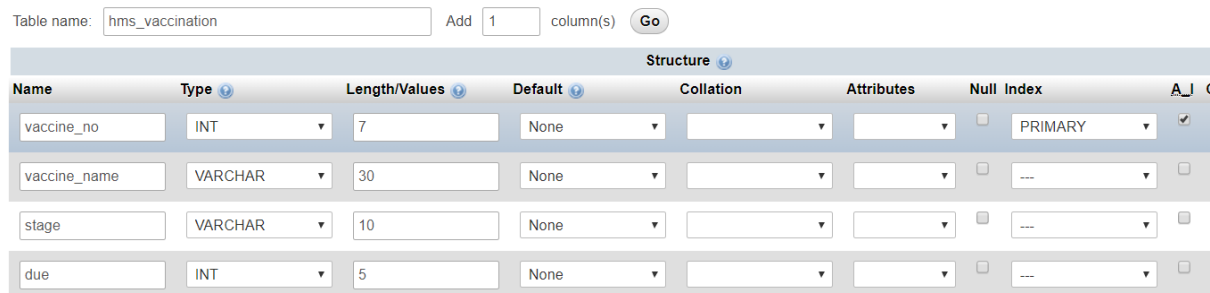


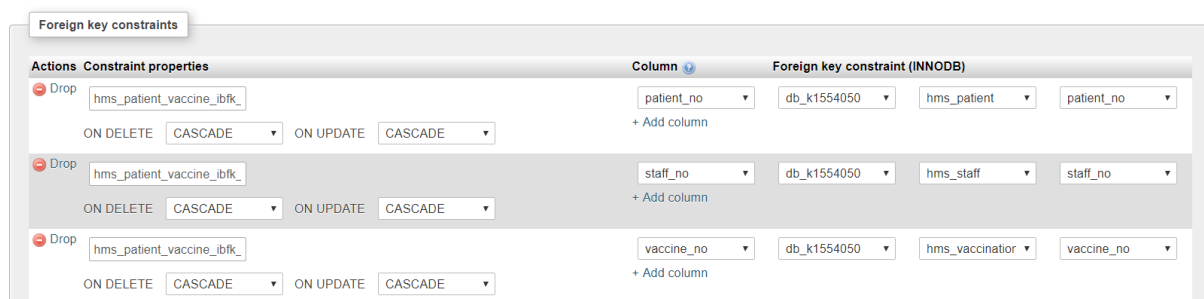
Table name: Add column(s)

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index
vaccine_no	INT	7	None			<input type="checkbox"/>	PRIMARY
vaccine_name	VARCHAR	30	None			<input type="checkbox"/>	---
stage	VARCHAR	10	None			<input type="checkbox"/>	---
due	INT	5	None			<input type="checkbox"/>	---

Figure 5.5 - Creation of Vaccination Table

particular error.

It was important to create relationships between tables in order to develop the required functionalities in the system. First of all, the particular foreign key column was defined as an index. The relationship between certain attributes can then be created through the relation view of the application. The 'On Delete' and 'On Update' were set to Cascade. Therefore, any update or delete changes there are made on the parent table will automatically execute the specific changes on the matching rows in the child table. This ensures that data is consistent throughout the application and



Foreign key constraints

Actions	Constraint properties	Column	Foreign key constraint (INNODB)
<input type="button" value="Drop"/>	<input type="text" value="hms_patient_vaccine_ibfk_1"/> ON DELETE: <input type="button" value="CASCADE"/> ON UPDATE: <input type="button" value="CASCADE"/>	<input type="text" value="patient_no"/> <input type="button" value="+ Add column"/>	<input type="text" value="db_k1554050"/> <input type="text" value="hms_patient"/> <input type="text" value="patient_no"/>
<input type="button" value="Drop"/>	<input type="text" value="hms_patient_vaccine_ibfk_1"/> ON DELETE: <input type="button" value="CASCADE"/> ON UPDATE: <input type="button" value="CASCADE"/>	<input type="text" value="staff_no"/> <input type="button" value="+ Add column"/>	<input type="text" value="db_k1554050"/> <input type="text" value="hms_staff"/> <input type="text" value="staff_no"/>
<input type="button" value="Drop"/>	<input type="text" value="hms_patient_vaccine_ibfk_1"/> ON DELETE: <input type="button" value="CASCADE"/> ON UPDATE: <input type="button" value="CASCADE"/>	<input type="text" value="vaccine_no"/> <input type="button" value="+ Add column"/>	<input type="text" value="db_k1554050"/> <input type="text" value="hms_vaccinator"/> <input type="text" value="vaccine_no"/>

Figure 5.2 - Creation of Foreign Keys on Patient Vaccine Table

is up to date.

All the necessary tables and foreign keys were created using the PhpMyAdmin's web application. This ensures that only the required data is stored in the database.

<input type="checkbox"/>	hms_appointment	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_availability	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_consultation	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_gp	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_other_md	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_patient	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_patient_vaccine	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_prescription	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_repeat	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_request	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_staff	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_users	★		Browse		Structure		Search		Insert		Empty		Drop
<input type="checkbox"/>	hms_vaccination	★		Browse		Structure		Search		Insert		Empty		Drop

Figure 5.3 - Creation of All Tables

5.3.2 Data Population

The database was then populated using dummy data. This data was generated using Excel and these files were formatted as 'OpenDocument Spreadsheet.' The reason for this is that it allows values to be defined as 'NULL' instead of empty strings. Therefore, CSV files were not used as it imports values as empty strings which may cause problems when importing data into the database such as 'date' values. This feature is useful for adding large amounts of data into the database as it will be quick and easy to perform. On the other hand, the web application only allows the user to add one record at a time. This will only be useful for testing data when implementing the functionalities in the system. Few records of the vaccinations table are shown on Figure 5.5 which shows the imported data from the OpenDocument file (Figure 5.4) with NULL values.

	A	B	C
1	vaccine_name	stage	due
2	Diphtheria	1	4
3	Diphtheria	2	4
4	Diphtheria	3	
5	Diphtheria	B	
6	Hepatitis A	1	6
7	Hepatitis A	2	6
8	Hepatitis A	3	
9	Hepatitis A	B	
10	Pertussis	1	4
11	Pertussis	2	4
12	Pertussis	3	
13	Pertussis	B	
14	Rotavirus	1	6
15	Rotavirus	2	

Figure 5.4 - Vaccinations Data (OpenDocument Spreadsheet)

	vaccine_no	vaccine_name	stage	due
<input type="checkbox"/>		1 Diphtheria	1	4
<input type="checkbox"/>		2 Diphtheria	2	4
<input type="checkbox"/>		3 Diphtheria	3	NULL
<input type="checkbox"/>		4 Diphtheria	B	NULL
<input type="checkbox"/>		5 Hepatitis A	1	6
<input type="checkbox"/>		6 Hepatitis A	2	6
<input type="checkbox"/>		7 Hepatitis A	3	NULL
<input type="checkbox"/>		8 Hepatitis A	B	NULL
<input type="checkbox"/>		9 Pertussis	1	4
<input type="checkbox"/>		10 Pertussis	2	4
<input type="checkbox"/>		11 Pertussis	3	NULL
<input type="checkbox"/>		12 Pertussis	B	NULL
<input type="checkbox"/>		13 Rotavirus	1	6
<input type="checkbox"/>		14 Rotavirus	2	NULL
<input type="checkbox"/>		15 Shingles	NULL	NULL

Figure 5.5 - Vaccinations Table in Database

5.3.3 Database Connection

The database connection was made on a single file where it includes parameters which contains the database name, username, password and the location of the database. The username and password are hidden for security purposes. The database was connected using a PDO connection. If connection was unsuccessful, an error message would be shown. The connection file is shown on Figure 5.6.

```
try {

    $pdo = new PDO(
        "mysql:host=localhost;dbname=db_k1554050", '', '');

} catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
}
```

Figure 5.6 - Database Connection File

5.3.4 Database Queries

Many complex queries were used to retrieve specific data from the database. Importantly, the use of inner joins was necessary as it will combine the information from the necessary tables based on the relationships between them. However, different types of joins and conditional statements were also used to retrieve specific data. The queries were first tested in PhpMyAdmin as it would be easier to identify any errors in the SQL statements. An example of select query can be found on figure 5.7 and this query retrieves all the vaccines that were given to a particular patient. It could be seen that this

query consists of three tables that are joined together to obtain this information. More examples of these SQL statements can be found in Appendix C.

```
1 select *, DATE_FORMAT(date, '%d/%m/%Y') as date, date as dates, COALESCE(DATE_FORMAT(hms_patient_vaccine.due, '%d/%m/%Y'), '-') as due,
2 first name as staff
3 from hms_patient_vaccine
4 inner join hms_staff
5 on hms_staff.staff_no = hms_patient_vaccine.staff_no
6 inner join hms_users
7 on hms_users.user_id = hms_staff.user_id
8 inner join hms_vaccination
9 on hms_patient_vaccine.vaccine_no = hms_vaccination.vaccine_no
10 where hms_patient_vaccine.patient_no = 2
11 order by dates DESC
```

Figure 5.7 - list of vaccinations given to a patient

5.4 Implementation of Key Functions

There were several functions that were developed for the system as it was considered as necessary for users. This also ensures that the system is fully functional and reliable to use. The development of some of these key functions will be discussed in this section.

5.4.1 Login Function

The login function can be considered be as the most important functionality in this system. This deals with data security as well as redirecting users to specific pages depending on their user type. The partial code for login can be seen below.

```
if(isset($_POST['login']))
{
    $username = $_POST['username'];
    $password = sha1($_POST["password"]);

    $sql = 'SELECT * FROM hms_users WHERE username = :username AND password = :password';
    $query = $pdo->prepare($sql);

    $query->execute(array(':username' => $username, ':password' => $password));

    if($query->rowCount() == 0){
        $_SESSION['error'] = '<div class="alert alert-danger"><strong>'. 'Incorrect username or password!'. '</strong></div>';
    }
    else{
        $row = $query->fetch(PDO::FETCH_ASSOC);

        session_regenerate_id();
        $_SESSION['sess_user_id'] = $row['user_id'];
        $_SESSION['sess_username'] = $row['username'];
        $_SESSION['sess_usertype'] = $row['user_type'];

        if( $_SESSION['sess_usertype'] == "Receptionist"){
            header('Location: receptionisthome.php');
        }
    }
}
```

Figure 5.7 - Login Code

The username and the password that was entered on the form are stored into variable. They are checked in the database to verify that the login details are valid. An error message would be displayed if the login details are invalid. If the details are valid, session variable would be created for

the 'user_id,' and 'user_type.' Session variables will allow this information to be used throughout the system. This can be useful when retrieving data about a specific user.

The user will then be redirected to a particular page depending their user type. Different user types have access to different pages. This function allows to create a single login page which is useful for this application and its users.

5.4.2 Adding Availability Slots

One of the tasks of receptionists is to add availability slots in order to create appointments later on. In this web application, there are two options of creating these slots. The user has an option to create a single appointment slot or multiple appointment slots. The user will fill in the required details on the form which will then be submitted.

The figure shows two web forms for creating appointment slots. The first form, titled 'Create single appointment slot', includes input fields for 'Date' (with a calendar icon), 'Time' (with a dropdown), 'Type' (with a dropdown), 'Clinician' (with a dropdown), and 'Location' (with a dropdown), followed by a blue 'Submit' button. The second form, titled 'Create multiple appointment slots', includes input fields for 'Date' (with a calendar icon), 'End Date' (with a calendar icon), 'From' (with a dropdown), 'To' (with a dropdown), 'Duration' (with a dropdown and 'mins' label), and 'Type' (with a dropdown), followed by a blue 'Submit' button.

Figure 5.8 - Create availability slots form

In order to create the function for creating multiple slots, the 'Date Period' class was used. Firstly, the dates that were chosen by the user were created into DateTime objects. The date interval was then created to support the DateTime class. This was created using the duration time (mins) that was chosen by the user.

The first Date Period object was created for a list of times which consists of the start time and the end time which are also chosen by the user. This will increment by the duration time. For example, if the user chose 9:00 as the start time, 9:30 as the end time and 15 mins for duration. This Date Period object will contain three times which are: 9:00, 9:15 and 9:30. Similarly, the second Date Period object was created for the list of dates. The process can be seen below on Figure 5.8.

```
$date = DateTime::createFromFormat('d/m/Y', $_POST['date']);

$begin = new DateTime($_POST['timeonehse'].":".$_POST['timeonemse']);
$endw = new DateTime($_POST['timetwohse'].":".$_POST['timetwomse']);

$interval = new DateInterval('PT'.$_POST['duration'].'M');
date_add($endw, $interval);

$times = new DatePeriod($begin, $interval, $endw);

$datetwo = DateTime::createFromFormat('d/m/Y', $_POST['datesetwo']);

$intervaltwo = new DateInterval('P1D');
date_add($datetwo, $intervaltwo);

$dates = new DatePeriod($date, $intervaltwo, $datetwo);
```

Figure 5.9 - Date Periods

Afterwards, an if statement was created to check if any one of availability slots exists in the database. If it exists, an error message will be shown to the user. On the other hand, availability slots

will be created for each date with the specific time if it does not exist. This will avoid the user creating duplicate slots which can cause problems when booking appointments for patients. It is also easier for creating several slots at once. This is beneficial for the user as they do not need to create appointment slots once at a time.

5.2.3 Booking Appointments

Appointments can be booked by receptionists and patients. However, patients are restricted to booking only online appointments. Receptionists can book appointments using two different approaches. They could navigate to the 'Current Day Appointments' page and can book any of the available appointments. Another approach is for them to book an appointment that is listed in 'Available Appointments.' However, both approaches follow similar steps to booking appointments. They click on the 'Book' button for the required appointment slot which will then display a modal. This requires them to enter a valid reason and to select a patient. A success message will then be shown to the user to notify them that the appointment is booked successfully.

Figure 5.10 - Appointment Booking Modal

In order for this functionality to work, jQuery and Ajax technologies were used. It also allows the user to see the updated content without refreshing the page. Once the user books the appointment, this availability slot the list of available information is refreshing the jQuery and Ajax were show specific on the modal. The process can be seen

```
$(document).on('click', '.book', function(){
    var id = $(this).attr("id");
    var action = "Selecttwo";
    $.ajax({
        url:"reccurappaction.php",
        method:"POST",

        data:{id:id, action:action},
        dataType:"json",
        success:function(data){
            $('#bookModal').modal('show');

            $('#action2').val("Book");
            $('#avaapp_id').val(id); //define id variable to hidden field
            $('#date').html(data.date);
            $('#time').text(data.time);
            $('#clinician').text(data.clinician);
            $('#reason').val("");
            $('#type').html(data.type);
            $('#myTable :checked').prop('checked', false);
            $('#myInput').val(null);
            $('#myTable tr').show();
            $('#result2').animate({scrollTop: "0px"});
            // $('#bookModal')[0].reset();
        }
    });
});
```

Figure 5.11 - jQuery Code for showing appointment details

will not be shown on appointments. This displayed without particular page. also used in order to appointment details jQuery code for this on Figure 5.11.

This modal will be displayed when they click on 'Book' button for an appointment slot. However, the information that is displayed is based on the id attribute that was defined on the selected button. This id refers to the appointment_id of the selected appointment. The id and the action variables are then sent to the reccurrappaction.php to access certain information. Therefore, an Ajax request will be sent using the POST method. The action variable is defined as 'SelectTwo' and this will be seen in an if statement on the particular page (Figure 5.12) to check if the variable value is set. In addition,

```
if($_POST["action"] == "Selecttwo")
{
    $output = array();
    $statement = $pdo->prepare(
        "SELECT DATE_FORMAT(date, '%d/%m/%Y') as date, TIME_FORMAT(time, '%H:%i') as time, a.first_name as clinician, type
        FROM hms_availability
        inner join hms_staff
        on hms_availability.staff_no = hms_staff.staff_no
        inner join hms_users a
        on a.user_id = hms_staff.user_id
        WHERE availability_no = '".$_POST["id"]."'"
        LIMIT 1"
    );
    $statement->execute();
    $result = $statement->fetchAll();
    foreach($result as $row)
    {
        $output["date"] = $row["date"];
        $output["time"] = $row["time"];
        $output["clinician"] = $row["clinician"];
        $output["type"] = $row["type"];
    }
    echo json_encode($output);
}
```

Figure 5.16 – reccurrappaction.php

the data that is returned from that page will be in Json format as it was stated as the datatype

The query on this Figure 5.12 will then be executed to obtain information about the particular appointment. The results of this query will be iterated using a foreach loop and the particular data will be added to the array called \$output. The data objects in the array will then be converted to a string that will be encoded in a JSON format. This will be then returned as a string containing the JSON representations of the particular values. Each value is then accessed through the data variable which can be seen on Figure 5.11 and are set to the particular html attributes. Therefore, this will show specific data on the modal. This helps the user to ensure that they selected the correct appointment slot. The checked fields on the patients table and the 'reason' input field on the modal will always be set to null. This prevents it from displaying previously entered data and checked values by the user.

The previous process showed how appointment details are shown on the modal. Next, the process of booking an appointment will be discussed. This process is

The jQuery code for shown below.

```
$('#action2').click(function(){
    var patientno = $('#myTable :checked').val();
    var reason = $('#reason').val();
    var booked = $('#booked').val();
    var id = $('#avaapp_id').val();
    var action = $('#action2').val();
    if(reason != '' && $('#input.patientcheck').is(':checked'))
    {
        $.ajax({
            url : "recurrappaction.php",
            method:"POST",
            data:{patientno:patientno, reason:reason, booked:booked, id:id, action:action},
            success:function(data){
                alert(data);
                $('#bookModal').modal('hide');
                fetchUser();
            }
        });
    }
    else
    {
        alert("Error! Please check data.");
    }
});
```

Figure 5.13 - jQuery Code for Booking Appointment

The patient number, reason, the checked value of the table (patient number), the booked value (staff number) and the value of the action will be sent as parameters in this ajax request. The value of the action is 'Book' as it defined using jQuery when displaying the booking modal. All of these variables are necessary for updating information on the database.

```
if($_POST["action"] == "Book") {
    $sql = "SELECT COUNT(availability_no) AS num FROM hms_appointment WHERE availability_no = :availability_no";
    $stmt = $pdo->prepare($sql);
    $stmt->bindValue(':availability_no', $_POST['id']);
    //Execute.
    $stmt->execute();
    //Fetch the row.
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    if(!$row['num'] > 0){
        $statement = $pdo->prepare(
            "INSERT INTO hms_appointment (availability_no, booked_by, reason, patient_no)
            VALUES (:availability_no, :booked_by, :reason, :patient_no)");
        $result = $statement->execute(
            array(
                ':availability_no' => $_POST['id'],
                ':booked_by' => $_POST['booked'],
                ':reason' => $_POST['reason'],
                ':patient_no' => $_POST['patientno']
            )
        );
    }
}
```

Figure 5.14 - Server Code for booking appointment

Figure 5.14 shows the partial server code for booking appointments. An appointment cannot be booked twice on the web application. However, it was decided to check this appointment slot in the server code to ensure that it is available. If it is available, the insert SQL statement would be executed which will insert a new appointment record in the hms_appointments table. This can be seen on Figure 5.14. Next, an update statement would also be executed to update the availability value to 'No' in the hms_availability table. Therefore, this particular appointment slot will not be shown on the list of available appointments. An error message would then be shown back to user as an alert using jQuery.

5.2.4 Requesting Repeat Prescriptions

Patients are able to request repeat medication if the clinicians decided to make it available to them. This can be seen as a list of repeat medications along with another list of their recent requests. They are able to select more than one repeat medication if needed for making requests. This can be useful feature for patient as it will be quicker and easier to requests for several repeat prescriptions. Once they made a request, this information will also be available to clinicians. Clinicians may approve or reject this request. In addition, patients will be able to see the status of their recent requests that were made within a month. The repeat prescription information for a particular patient is shown on Figure 5.15.

Recent Requests

Request Date	Drug	Status
23/04/2018	Simvastatin 10mg	In Progress

Available Repeat Prescriptions

<input type="checkbox"/>	Last Issued	Drug	Dosage	Quantity	Available Requests
<input type="checkbox"/>	01/04/2018	Lisinopril	Once a day	50	4
<input type="checkbox"/>	24/02/2018	Simvastatin 10mg	Once a day	30	3

☒ Request

Figure 5.15 - Repeat Prescriptions Page

In order to create the request function, jQuery and Ajax was also used. When the user clicks on the 'Request' button, it will display a confirmation that ask the user to confirm the selected repeat prescriptions. Every checked value will be pushed into an array called 'id.' The checked values contain the 'repeat_id' for the specific data. If no checkboxes are selected, it will ask the user to select at least one checkbox. The array and the action variable called 'request all' will be sent to the corresponding page that deals with server. An insert statement is executed for each 'id' in the array and this inserts the selected requests into the database. A success message will then be shown to the user. This process can be seen in Figure 5.16 and 5.17.

```

$('#requestall').click(function(){

    if(confirm("Please confirm to request these selected repeat prescriptions!"))
    {
        var action = 'requestall';

        var id = [];

        $("input:checkbox[class=checkboxes]:checked").each(function(){
            id.push($(this).val());
        });

        if(id.length < 1)
        {
            alert("Please select at least one repeat prescription!");
        }
        else
        {
            $.ajax({
                url:"patpresaction.php",
                method:"POST",
                data:{id:id, action:action},
                success:function(data)
                {
                    fetchRequest();
                    fetchRepeat();
                    alert(data);
                }
            });
        }
    }
});

```

Figure 5.16 - jQuery Code for requests

```

$('#requestall').click(function(){

    if(confirm("Please confirm to request these selected repeat prescriptions!"))
    {
        var action = 'requestall';

        var id = [];

        $("input:checkbox[class=checkboxes]:checked").each(function(){
            id.push($(this).val());
        });

        if(id.length < 1)
        {
            alert("Please select at least one repeat prescription!");
        }
        else
        {
            $.ajax({
                url:"patpresaction.php",
                method:"POST",
                data:{id:id, action:action},
                success:function(data)
                {
                    fetchRequest();
                    fetchRepeat();
                    alert(data);
                }
            });
        }
    }
});

```

Figure 5.17 - Server code for requests


5.2.5 Adding

patients

The vaccinations
patients are

Add Vaccination

Vaccination Details:

Date:  Vaccination:

Stage:


Due Date (If applicable): 

Figure 5.18 - Adding vaccinations for patients form

vaccinations for

that are given to the
recorded on the

system. The details that are recorded includes the date, vaccination, stage and due date if applicable. The stage is dependent on the vaccination whereas the due date is dependent on the date and the stage. The stage and due date are generated according to what the user chooses. This information is recorded by clinicians. The form for entering the vaccinations for patients are shown below on Figure 5.18.

This function was mainly created using dependent dropdowns which also uses jQuery and Ajax. The function on Figure 5.19 will execute every time when a change is made on the vaccine field. This can be described as the time when the user chooses a vaccine from dropdown of vaccines as seen on the form. The vaccine name and the action called 'changevaccine' variables are sent to the corresponding page which deals with the server.

```
$('#advaccine').change(function(){
    var date = $('#adddatevaccinetext').val();

    if($(this).val() != '')
    {
        var action = 'changevaccine';
        var vaccine = $(this).val();

        $.ajax({
            url:"clpatientdetaddaction.php",
            method:"POST",
            data:{action:action, vaccine:vaccine},
            success:function(data){
                $('#addstage').html(data);
                $('#adddatedue').data('datepicker').setDate(null);
            }
        })
    }
});
```

Figure 5.19 - jQuery code of generating stages

```

if($_POST["action"] == "changevaccine")
{
    $sql = "SELECT vaccine_no, stage FROM hms_vaccination WHERE vaccine_name = '".$_POST["vaccine"]."' GROUP BY stage";

    $statement = $pdo->prepare($sql);
    $statement->execute();
    $result = $statement->fetchAll();
    $output = '';

    $output .= '<option selected disabled></option>';
    if($statement->rowCount() > 0)
    {
        foreach($result as $row)
        {
            if ($row["stage"] == NULL) {
                $stage = 'No Stage';
            }
            else {
                $stage = $row["stage"];
            }
            $output .= '<option value="'.$row["vaccine_no"].'">'.$stage.'</option>';
        }
    }

    echo $output;
}

```

Figure 5.20 - Server code of generating stages

On the server side, the vaccine number and the stages will be selected for the particular vaccine using a SQL query. If there are no stages for that vaccine then the string that will be sent back will say 'No stage.' If stages exist for this vaccine, the list of stages will be sent as options for the dropdown. The vaccine number will represent the certain stages. This data will then be accessed using jQuery and it will set the 'stage' dropdown with the options that were sent. Similarly, the due date is generated using the method for showing stages on the form. This can be found in Appendix D.

5.2.6 Sending Vaccination Reminders

Receptionists are able to send reminders to patients who have vaccinations that are due soon. They have to option to filter the due dates before they are sent. This avoids sending reminders too early for patients. In addition, they also can send multiple vaccination notifications or they can send it to a particular patient. The patient will receive this reminder by email where it would state their vaccination is due soon and they should book an appointment with the GP. The interface for this functionality can be seen on Figure 5.21.

Filter: From <input type="text"/>		To <input type="text"/>	<input type="button" value="Search"/>	<input type="button" value="Clear Filter"/>
<input type="text" value="Search.."/>				
<input type="checkbox"/>	Patient	Vaccination	Due Date	
<input type="checkbox"/>	Harry Mills 30/01/1999	Pertussis	21/05/2018	<input type="button" value="✓ Send"/>
<input type="checkbox"/>	Kelly Rogers 10/07/1982	Hepatitis A	04/06/2018	<input type="button" value="✓ Send"/>
<input type="button" value="✓ Send Multiple"/>				

Figure 5.21 - Patients' Vaccinations and their due dates

This functionality was developed with the use of PHPMailer library. This library is well known for sending emails therefore it was used for this project. This library has support for SMTP which allows the system to send emails without the need for a local server. An email delivery service called 'SendGrid' was used for SMTP authentication. Therefore, the SMTP was configured in the PHPMailer file with details such as username, password, server host name and the port number. These details are shown on Figure 5.22. Username and Password is hidden for security purposes.

```
$mail = new PHPMailer(true);
try {
    //Server settings
    $mail->SMTPDebug = 2;
    $mail->isSMTP();
    $mail->Host = 'smtp.sendgrid.net';
    $mail->SMTPAuth = true;
    $mail->Username = '';
    $mail->Password = '';
    $mail->SMTPSecure = 'ssl';
    $mail->Port = 465;
```

Figure 5.22 - SMTP Authentication

The user will select the required patient vaccines for reminders. This will then be stored in an array which will be pushed to the PHPMailer file. Emails will be sent for each patient that is the array with their names that was obtained using SQL queries. The content that will be sent to the users is shown below on Figure 5.23.

```
$mail->isHTML(true); // Set email format to HTML
$mail->Subject = 'Vaccination Reminder';
$mail->Body = 'Hello '.$name.', <br><br>
We would like to remind you that you have a vaccine due soon. <br>
Please book an appointment with the GP as soon as possible.<br><br>
Thank you.<br>
HealthCare GP.';
$mail->AltBody = 'We would like to remind you that you have a vaccine due soon.
Please book an appointment with the GP as soon as possible. Thank you. HealthCare GP.';
$mail->send();
```

Figure 5.23 - Email Content

This will also be updated in the database where it would state the particular patients are notified. The SQL query for this update is shown on Figure 5.24.

```

$statement = $pdo->prepare(
    "UPDATE hms_patient_vaccine
    SET notification = :notify
    WHERE pv_no = :id
    "
);
$result = $statement->execute(
    array(
        ':notify' => 'Sent',
        ':id' => $id
    )
);

```

Figure 5.24 - Update SQL Query

5.3 Security Concerns

Importance of security has been emphasized on this system as it deals with confidential and sensitive information. Username and password provided authentication as unauthorized users cannot access the data without these details. The password has also been encrypted using the SHA1 encryption method which can be seen on Figure 5.25.

```

$password = sha1($_POST['password']);

```

Figure 5.25 - SHA1 Encryption

Users cannot register to the system immediately and this prevents any authorized users being able to access the services on the system. Registration can only be done with the use of patient or staff number and an access code. These details will be sent to the required user if requested. However, the access code can only be used once and it will expire in four weeks. This increases the security of the system.

In addition, all pages in the system are protected as it will redirect to the login page when the specific URL of certain pages are entered. Only certain users can access certain page. An example of this code for this protection is shown below.

```

$usertype = $_SESSION['sess_usertype'];
if(!isset($_SESSION['sess_username']) || $usertype!="Receptionist"){
    header('Location: login.php');
}

```

Figure 5.26 - Protected Pages

5.4 Conclusion

This implementation chapter helped to gain knowledge about the development of this application using different technologies and tools. The application has been developed successfully as all requirements of the system are met. Technologies such as Ajax was used to improve the usability of the system.

Improvements could be made to the system such as the generating files using the data from the database. This would be useful for sending letters or medical details to healthcare organisations.

Testing – Chapter 6

6.1 Functional Testing

Functional testing is a software testing process in which the system is tested against all the functional requirements. Functional tests were carried out initially during development and this testing was then continued throughout the software development cycle. These tests are important to ensure that each feature works correctly and functions appropriately to all end users of the system. One of the techniques that was used during functional testing was Black Box Testing. This type of testing does not require the knowledge of the internal structure of the system or how it works. It examines the functionalities at a user interface level by evaluating the outputs with specific inputs. Therefore, several test cases were built around the requirements and each one of them was provided with an expected output for a particular input.

ID	FR ID	SCENARIO	EXPECTED RESULT	RESULT
1	1.1	<ul style="list-style-type: none">- User clicks 'users' link.- User clicks on Add button.- User fill in required details.- User submits	Success message displaying that the patient/staff is added successfully.	PASS
2	1.2	<ul style="list-style-type: none">- User clicks on 'Users' link.	List of users is displayed.	PASS
3	1.3	<ul style="list-style-type: none">-User clicks on the 'Disable' button for the required user.	Message displaying that the access has been for this user has been disabled.	PASS
4	2.1	<ul style="list-style-type: none">- User clicks on 'Patient' link.- User selects the patient.- User choose an option from the 'add' dropdown button.-User fill in required details.-User submits the data.	Success message will be displayed according to the patient details that has been added.	PASS
5	2.2	<ul style="list-style-type: none">-User clicks on 'Patient' link.-User selects the patient.	List of patient details are displayed.	PASS
6	2.3	<ul style="list-style-type: none">-User clicks on 'Patient' link.-User selects patient. <p>User clicks on update button of the required details that needs to be change.</p>	Success message will be displayed stating that the details has been updated.	PASS
7	3.1	<ul style="list-style-type: none">-User clicks on 'Availability' link.-User fill in required details.	Success message will be displayed saying that the appointment slots has been	PASS

		-User submits form.	added successfully.	
8	3.2	-User clicks on 'Availability'	List of appointments will be shown.	PASS
9	3.3	-User clicks on 'Appointments' -User clicks on the delete button for the required availability/appointment slot.	Message will be shown saying the appointment slot has been deleted.	PASS
10	4.1	-User clicks on 'Appointments'	List of appointments will be shown.	PASS
11	4.2	-User clicks on 'Appointments' -User clicks on the button 'Book' for the required appointment.	Success message will be shown saying that the appointment is booked successfully.	PASS
12	4.3	-User clicks on 'Appointments.' -User enters the specific detail in the search box.	List of filtered data will be shown.	PASS
13	4.4	-User clicks on 'Appointments' -User clicks on the 'cancel' button for the required appointment.	Message will be showing saying that the appointment is cancelled successfully.	PASS
14	4.5	-User clicks on 'Appointments.' -User clicks on the 'update' button of the required appointment. -User fills in required details. User submits the data.	Success message saying the appointment is updated successfully.	PASS
15	5.1	-User clicks on 'Patients' -User selects the patient -User choose the 'prescription' option from 'add' dropdown button. -User fill in required details. -User submits the data.	Success message displayed saying that the data is updated.	PASS
16	5.2	-User clicks on 'Patients' -User selects the patient	A list of prescription details would be displayed.	PASS
17	5.3	-User clicks on 'Prescriptions' -User selects the required repeat medication. -User clicks on the 'Request' button.	A success message will be shown saying that the requests are successful.	PASS
18	5.4	-User clicks on 'Prescriptions'	A message will be shown saying	PASS

		-User 'approve' or 'reject' the required request using buttons.	that the request is approved or rejected.	
19	6.1	-User clicks on 'Vaccinations' -User clicks on the 'Add' button. -User fill in required details. -User submits data.	A success message will be shown saying that the vaccinations is added successfully.	PASS
20	6.2	-User clicks on 'Vaccinations' -User clicks on 'update' button for the required vaccination. -User fill in required details. -User submits data.	A success message will be shown saying that the vaccination is updated successfully.	PASS
21	6.3	-User clicks on 'Vaccinations' and then clicks on 'Vaccination Notifications' -User selects the required patient vaccinations. -User can either click 'send' or 'send all' button.	A success message will be shown saying that the notifications are sent.	PASS

All tests have been passed and it can be said that this system is fully functional and accessible to all users.

Critical Review – Chapter 7

The system has been developed according to the design and have met the requirement that were gathered in the analysis phase. All tests were passed therefore it can be considered that users are able to access the services they require. Some functionalities that implemented can be considered a quite challenging such as handling prescription data. Importance was made on data security as confidential and sensitive data were being handled by the system.

References

Room, S. (2007). Data protection and compliance in context. Swindon: British Computer Society, p.1.

ICO (2018). Key definitions of the Data Protection Act. [online]
Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/key-definitions/>
[Accessed 17 Feb. 2018].

BBC (2018). Registration with the Information Commissioner. [online]
Available at: <http://www.bbc.co.uk/schools/gcsebitesize/ict/legal/0dataprotectionactrev3.shtml>
[Accessed 17 Feb. 2018].

Roebuck, W. (2018). Complying with the Data Protection Act. [online]
Available at: <https://www.icaew.com/en/technical/information-technology/it-management/complying-with-the-data-protection-act>
[Accessed 17 Feb. 2018].

Walker, D. and Millman, R. (2018). Data protection principles. [online]
Available at: <http://www.itpro.co.uk/data-protection/28020/data-protection-principles>
[Accessed 17 Feb. 2018].

GOV (2018). Data protection. [online]
Available at: <https://www.gov.uk/data-protection>
[Accessed 17 Feb. 2018].

ICO (2018). The rights of individuals (Principle 6). [online]
Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/principle-6-rights/>
[Accessed 17 Feb. 2018].

Jay, R. and Clarke, J. (2010). Data Protection Compliance in the UK: A Pocket Guide. IT Governance Ltd., pp.18-24.

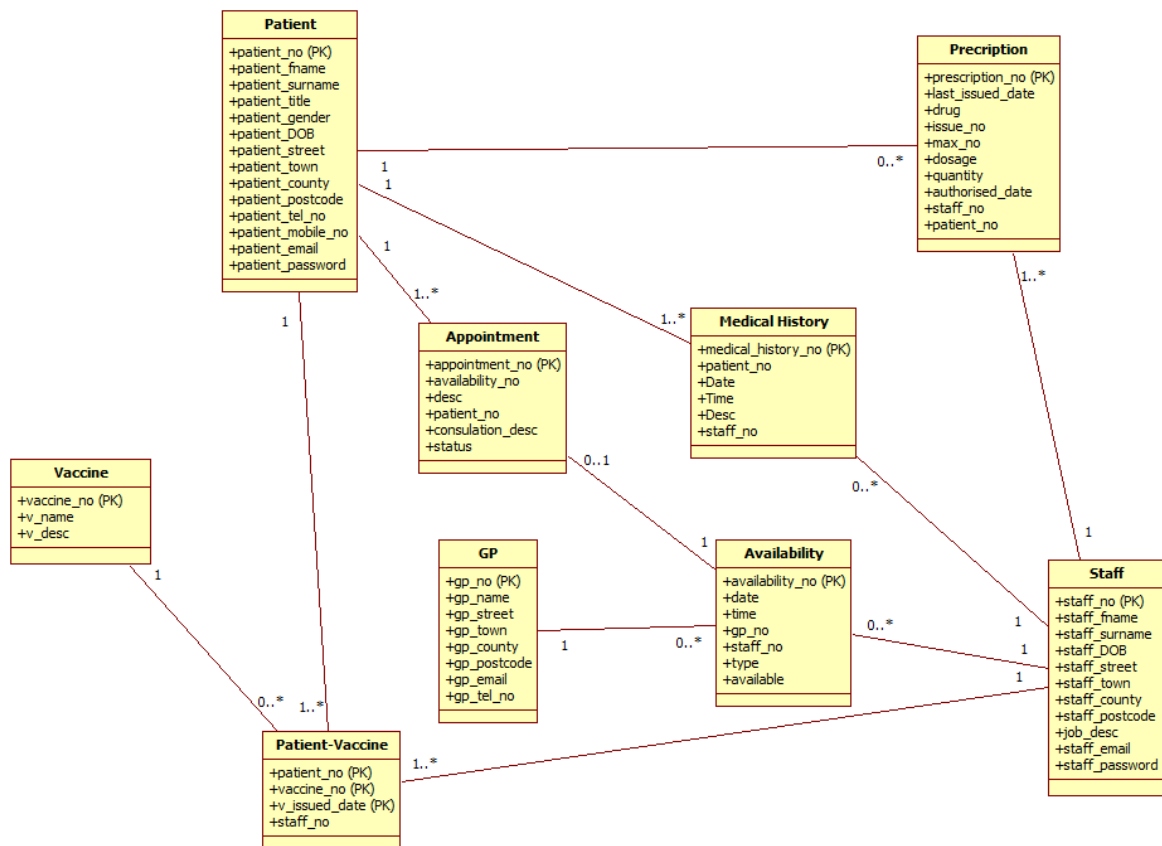
Kumar, V., Srivastava, J. and Lazarevic, A. (2005). Managing Cyber Threats. p.4.

Schulman, A (2018). Top 10 database attacks | BCS - The Chartered Institute for IT. [online]
Available at: <https://www.bcs.org/content/ConWebDoc/8852>
[Accessed 25 Feb. 2018].

Huang, L. (2013). Research and Application of Web Database Security Technology. Applied Mechanics and Materials, 380-384, p.2307.

Appendices

Appendix A – Initial ER diagram



Appendix B – Wireframes

HealthCare GP

RegisterLogin

Patient Registration

Register

Patient No

Access Code

Username

Password

Confirm Password

Submit

Copyright © 2018 HealthCare GP

HealthCare GP

HomeAppointmentsAvailabilityPatientsVaccinationsUsers

My AccountLogout

Availability

Add Availability

Create Single Appointment Slot

Date:Time:Type:Clinician:Location:Submit

Create Multiple Appointment Slots

Date:End Date:From:To:Duration:Type:Clinician:Location:Submit

Home | Appointments | Availability | Patients | Vaccinations | Users

Copyright © 2018 HealthCare GP

HealthCare GP

HomeAppointmentsAvailabilityPatientsVaccinationsUsers

My AccountLogout

Vaccinations

Vaccination Notifications

Add

Search

Name

Stage

Due (weeks)

Update

Delete

Home | Appointments | Availability | Patients | Vaccinations | Users

Copyright © 2018 HealthCare GP

Figure A - Wireframes

Appendix C

QUERIES

1. List all available appointments

```
select hms_availability.availability_no, hms_users.first_name as clinician, hms_gp.gp_name,
hms_availability.type, concat(DATE_FORMAT(hms_availability.date, '%d/%m/%Y'), ' ',
TIME_FORMAT(hms_availability.time, '%H:%i')) as `date`
FROM hms_availability
inner join hms_staff
on hms_availability.staff_no = hms_staff.staff_no
inner join hms_users
on hms_users.user_id = hms_staff.user_id
inner join hms_gp
on hms_gp.gp_no = hms_availability.gp_no
where hms_availability.available = 'Yes' and hms_availability.staff_no LIKE $clinician and
hms_availability.gp_no LIKE $location and hms_availability.type LIKE $type
order by hms_availability.date, hms_availability.time
```

2. Delete an appointment

```
DELETE FROM hms_appointment WHERE appointment_no = :id
```

3. List appointments for today's date of doctor/nurse

```
select hms_appointment.appointment_no, u.first_name as clinician, CONCAT(u1.first_name, '
',u1.surname, ' ', DATE_FORMAT(u1.DOB, '%d/%m/%Y')) as patient, hms_availability.type,
TIME_FORMAT(hms_availability.time, '%H:%i') as `time`, reason, status,
COALESCE(TIME_FORMAT(arrival_time, '%H:%i'), '-----') AS arrival_time,
hms_appointment.patient_no
FROM hms_appointment
INNER JOIN hms_availability
ON hms_availability.availability_no = hms_appointment.availability_no
inner join hms_staff
on hms_availability.staff_no = hms_staff.staff_no
inner join hms_users u
on u.user_id = hms_staff.user_id
inner join hms_patient
on hms_patient.patient_no = hms_appointment.patient_no
inner join hms_users u1
on u1.user_id = hms_patient.user_id
where hms_availability.date = CURDATE() and hms_availability.staff_no LIKE $staffno
order by hms_availability.date, hms_availability.time
```

4. List vaccinations

```
select *, COALESCE(stage, '-') as stage, COALESCE(due, '-') as due FROM hms_vaccination
```

5. List all repeat prescription requests

```
select CONCAT(u.first_name, ' ',u.surname, ' ', DATE_FORMAT(u.DOB, '%d/%m/%Y')) as patient,
DATE_FORMAT(request_date, '%d/%m/%Y') as request_date, DATE_FORMAT(last_issued,
'%d/%m/%Y') as last_issued, drug, r_issue_no, max_no, dosage, quantity, status, request_no,
u1.first_name as prescriber
from hms_request
inner join hms_repeat
on hms_request.repeat_no = hms_repeat.repeat_no
inner join hms_patient
on hms_repeat.patient_no = hms_patient.patient_no
```

```

inner join hms_users u
on hms_patient.user_id = u.user_id
inner join hms_staff
on hms_repeat.prescriber = hms_staff.staff_no
inner join hms_users u1
on hms_staff.user_id = u1.user_id

```

6. List repeat prescription history of patient (12 months)

```

select request_no, DATE_FORMAT(hms_request.request_date, "%d/%m/%Y") as date,
hms_repeat.drug, dosage, quantity, COALESCE(DATE_FORMAT(last_issued, "%d/%m/%Y"), "-") as
lastissued, hms_request.status
FROM hms_request
INNER JOIN hms_repeat
on hms_request.repeat_no = hms_repeat.repeat_no
where hms_repeat.patient_no = $patientno and hms_request.request_date <=
DATE_ADD(CURDATE(), INTERVAL 1 YEAR)
order by hms_request.request_date DESC

```

7. List medical history of patient

```

select rank, patient_no, date, dates, staff, description from (
select 1 as rank, hms_consultation.patient_no as patient_no, DATE_FORMAT(date, '%d/%m/%Y') as
date, date as dates, first_name as staff, consultation_desc as description from hms_consultation
INNER JOIN hms_staff
ON hms_staff.staff_no = hms_consultation.staff_no
INNER JOIN hms_users
ON hms_users.user_id = hms_staff.user_id
INNER JOIN hms_patient
on hms_patient.patient_no = hms_consultation.patient_no
UNION ALL
select 2 as rank, hms_prescription.patient_no as patient_no, DATE_FORMAT(hms_prescription.date,
'%d/%m/%Y') as date, hms_prescription.date as dates, u1.first_name as staff, CONCAT(drug, ' Issue
No: ', COALESCE(issue_no, '-'), ', ', Dosage: ' ', dosage , ', ', Quantity: ', quantity, ', ', Type: ' ', type) as
description
from hms_prescription
inner join hms_staff s
on s.staff_no = hms_prescription.prescriber
inner join hms_users u
on u.user_id = s.user_id
inner join hms_staff s1
on s1.staff_no = hms_prescription.staff_member
inner join hms_users u1
on u1.user_id = s1.user_id
where hms_prescription.status != 'Deleted' and hms_prescription.issue_no != 0
UNION ALL
select 3 as rank, hms_patient_vaccine.patient_no as patient_no, DATE_FORMAT(date, '%d/%m/%Y')
as date, date as dates, first_name as staff, CONCAT(vaccine_name, ' Stage: ', COALESCE(stage, '-'), '
Due Date: ', COALESCE(DATE_FORMAT(hms_patient_vaccine.due, '%d/%m/%Y'), '-')) as description
from hms_patient_vaccine
inner join hms_staff
on hms_staff.staff_no = hms_patient_vaccine.staff_no
inner join hms_users
on hms_users.user_id = hms_staff.user_id
inner join hms_vaccination

```

```

on hms_patient_vaccine.vaccine_no = hms_vaccination.vaccine_no
UNION ALL
select 4 as rank, hms_other_md.patient_no as patient_no, DATE_FORMAT(date, '%d/%m/%Y') as
date, date as dates, first_name as staff, CONCAT(type, ': ', md_desc) as description
from hms_other_md
inner join hms_staff
on hms_staff.staff_no = hms_other_md.staff_no
inner join hms_users
on hms_users.user_id = hms_staff.user_id ) a
where patient_no = $patientno

```

8. Update a vaccination

```

UPDATE hms_vaccination
SET vaccine_name = :vname, stage = :stage, due = :due
WHERE vaccine_no = :id

```



```

$('#addstage').change(function(){

    var date = $('#adddatevaccinetext').val();
    var action = 'changestage';
    var vaccine_no = $(this).val();

    if(date != '')
    {

$.ajax({
    url:"clipatientdetaddaction.php",
    method:"POST",
    data:{action:action, vaccine_no:vaccine_no, date:date},
    success:function(data){
        $('#adddatedue').datepicker('update', data);
    }
    })
    }
});

```

Figure D1 - jQuery code of adding due dates

```

if($_POST["action"] == "changestage")
{

    $date = DateTime::createFromFormat('d/m/Y', $_POST['date']);
    $last = $date->format('Y-m-d');

    $stmt = $pdo->prepare("SELECT due*7 as due FROM hms_vaccination WHERE vaccine_no = '".$_POST["vaccine_no"]." ' ");
    $stmt->execute();
    $data = $stmt->fetch();
    if ( $data['due'] > 0) {
        $due = $data['due'];

        $newdate = date('Y-m-d', strtotime($last. ' + '.$due.' days'));

        $newdate1 = DateTime::createFromFormat('Y-m-d', $newdate);
        $last = $newdate1->format('d/m/Y'); }
    else {
        $last = NULL;
    }

    echo $last;
}

```

Figure D2 - Server code of generating stages