Mental Health Management Platform

GROUP 2

REPORT

04/04/2024

OLUWASEUN AYEGBUSI
JJ BUZZARD
COSTAS HADJINEOPHYTOU
AN-TIEN HUANG
MATTHEW PENIKET
KEAN TOH

COMP0071: Software Engineering

Department of Computer Science
University College London

Executive Summary

A shortcoming of existing electronic health care record systems is that they lack the key functionality required in a mental health care context. The Mental Health Management Platform (MHMP) is a proposed web application designed to enhance the delivery and coordination of mental health services in the UK. The MHMP aims to bridge the gap between mental health professionals and patients by providing a user-friendly, efficient, and interconnected platform for communication and collaboration.

Key features of the MHMP include an appointment system, clinical notes accessibility, access to prescription information, a messaging system, and a user-friendly interface. The report outlines the functional and non-functional requirements of the MHMP system using the MoSCoW prioritisation method and presents a Unified Modelling Language (UML) use case diagram to visualise the interactions between the system and its actors.

An extensive domain analysis was conducted to identify and categorise system entities, actors, and their interrelations. The use case, analysis and design class, component, and deployment diagrams were created to represent the system's static structure, while sequence, state machine and activity diagrams were used to model the system's dynamic behavior.

The report also presents an application mock-up showcasing the MHMP's design, layout, and functionality for several core use cases, demonstrating a commitment to delivering a simple, usable, and optimal user experience for the diverse parties involved in mental health care within the UK.

In conclusion, the MHMP is a promising solution to address the critical need for integrated communication and collaboration within the UK's mental health sector. The comprehensive analysis, design, and mock-ups presented in the report lay a solid foundation for the future development and implementation of the MHMP, which has the potential to streamline operational processes, increase connection between, and improve, the mental health care experience for patients and professionals alike.

Contents

Chapter 1

Introduction and Problem Statement

1.1 Introduction

In the modern era, the healthcare sector, especially mental health services, requires technological innovation to help cope with the high level of patient demand it is subject to (**Figure 1.1.0.1**, **1.1.0.2**). The Mental Health Management Platform (MHMP) aims to bridge the gap between mental health professionals and those in need of their services. This web application, adaptable for both mobile and desktop usage, aims to enhance the delivery and coordination of mental health services delivered by mental health trusts within the UK, fostering an efficient healthcare experience for the diverse parties involved.

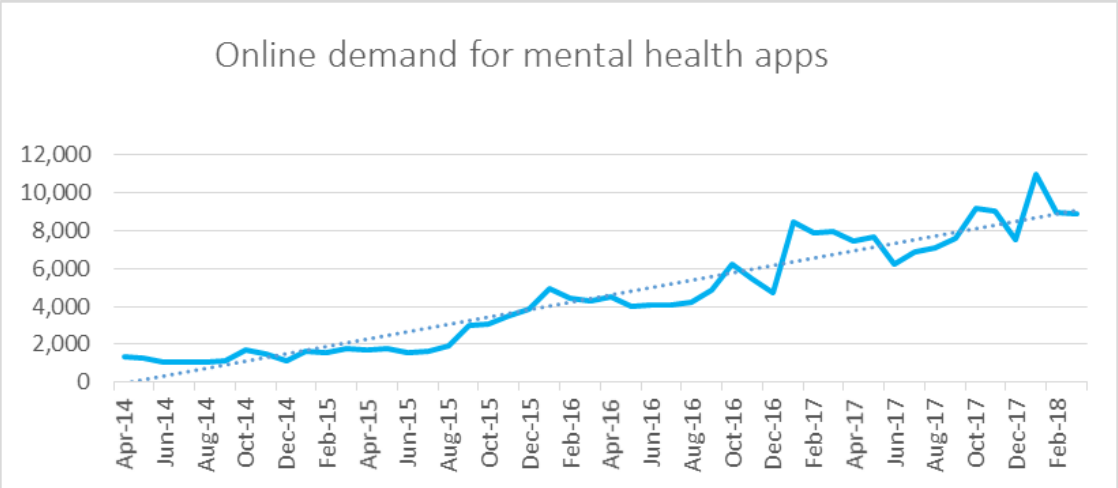


Figure 1.1.0.1: Demand for mental health applications, April 2014 - February 2018 [1]

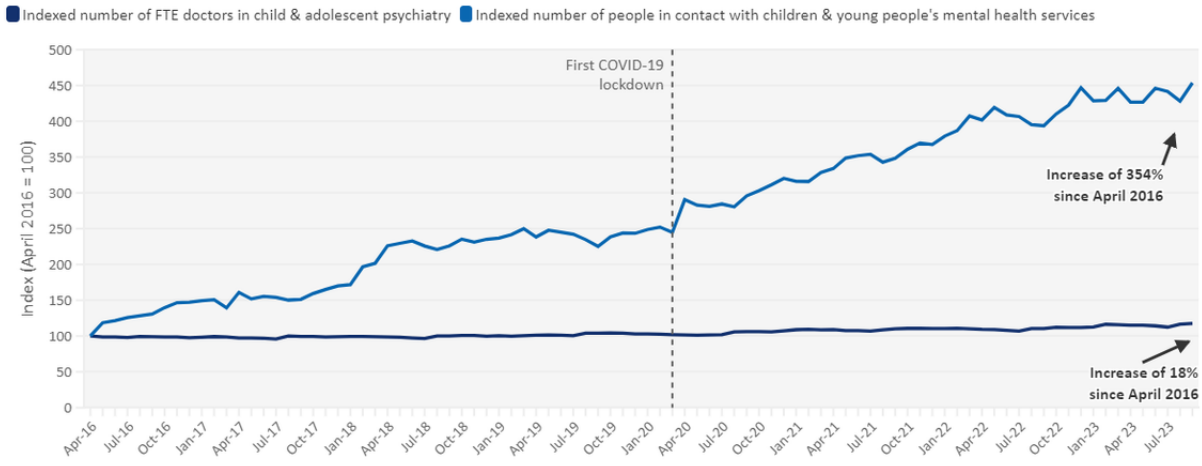


Figure 1.1.0.2: Child and adolescent mental health services: Indexed number of full-time equivalent doctors relative to the number of people in contact with services, 04/2016 - 10/2023 [2]

1.2 Project Brief

Through discussion, a problem was identified within the UK health sector - the lack of a dedicated application to cater for the needs of professionals and patients within the mental health system. The Mental Health Management Platform (MHMP) is the proposed solution. The original project brief is as follows:

> *The Mental Health Management Platform (MHMP) is a proposed web application for mobile and desktop aimed at enhancing the delivery and coordination of mental health services. Designed for patients, pharmacists, therapists / clinical psychologists, psychiatrists, and nurses / support workers, MHMP features a notification system for appointments, access to prescription information, and the ability for clinical staff to view and update clinical notes. With a focus on user-friendliness, and efficient healthcare delivery, MHMP addresses the critical need for integrated communication and collaboration within the mental health sector.*

1.3 Background and Purpose

Mental health care in the UK brings together a variety of different clinical professionals to deliver care for patients. This care takes place within individual mental health trusts, which deliver care to patients within a defined geographical region [3]. Mental health trusts comprise different types of services – inpatient hospitals and care in the community. It would benefit the diversity of professionals working in the sector and patients if there were a dedicated web application to streamline the communication and operational processes of mental health care.

We aim to target the application as a comprehensive solution for mental health trusts within the UK. By integrating features such as a notification system for appointments, access to prescription information, and the ability for clinical staff to view and update clinical notes, the MHMP addresses the critical need for an interconnected platform. This platform not only

promises to improve the workflow for healthcare providers but also ensures that patients receive timely and appropriate care.

1.4 Similar Application Review

While conducting research on existing mental health management applications, we identified several similar platforms currently or formerly in use within the UK healthcare system. These applications, while not always specifically tailored for mental health services, provide valuable insights into the features and functionalities that are essential for effective patient management and care coordination.

1.4.1 Lorenzo

Lorenzo was a widely used electronic health record system in the UK, designed to manage various aspects of patient care, including mental health services. Developed by iSOFT (later acquired by DXC Technology), Lorenzo aimed to provide a comprehensive, integrated solution for healthcare providers across different care settings. However, the implementation of Lorenzo faced several challenges, such as poor user experience, difficulty in customisation, and limited interoperability with other systems. These issues led to significant delays in deployment and increased costs for healthcare organisations. As a result, many NHS trusts opted to replace Lorenzo with alternative solutions, and the system is no longer in widespread use, and Lorenzo has been heavily criticised [4].

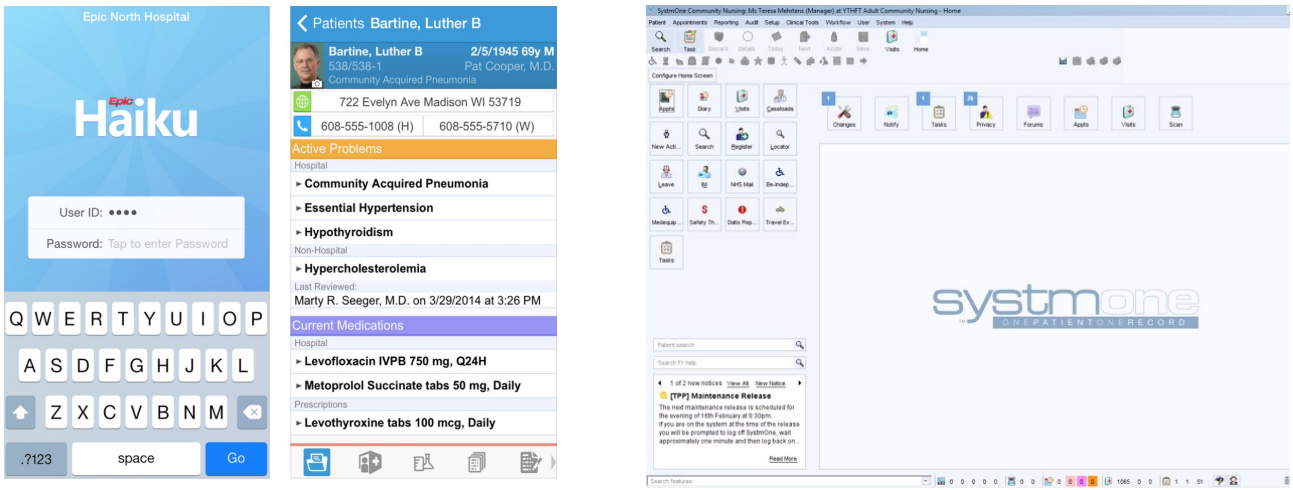
1.4.2 Epic

Epic is a comprehensive electronic health record (EHR) system commonly used by hospitals and healthcare organisations in the UK and worldwide [5]. Developed by Epic Systems Corporation, it offers a wide range of features. While Epic is a powerful tool for general healthcare, it may not be specifically tailored to the unique needs of mental health services. Additionally, the complexity of the system and the significant investment required for implementation and maintenance may pose challenges for some mental health organisations. Epic includes a messaging system, which we are adopting within our application. Epic also has a mobile app (Epic Haiku, see **Figure 1.4.3.1a**). These two features are ideal for the diverse needs of those working in the mental health sector, who may be visiting patients receiving care in the community, necessitating a platform for timely communication between those working in different environments. Further, from discussion with a recognised shortcoming of existing EHR systems is that they

1.4.3 SystmOne

SystmOne is an electronic health record system widely used by GP services, community health providers, and mental health trusts in the UK developed by TPP (The Phoenix Partnership). SystmOne provides a range of features. While SystmOne is well-suited for primary care and has been adopted by some mental health trusts, it may not offer all the specific functionalities required for specialised mental health services. The system's focus on general practice means that certain features, such as appointment reminders, separa-

tion between psychiatric and therapist notes, and secure messaging between mental health professionals, may be limited.



(a) Epic Haiku Mobile User Interface

(b) SystmOne Desktop User Interface

Figure 1.4.3.1: Comparison of User Interfaces

1.5 Our Application and Report

1.5.1 The Need for a Dedicated Mental Health Management Platform

Although the applications discussed above have contributed to the digitalisation of healthcare services and have been adopted to varying degrees by mental health organisations, there remains a clear need for a dedicated platform that caters specifically to the requirements of mental health professionals and patients [6]. The Mental Health Management Platform (MHMP) aims to fill this gap by providing a comprehensive, user-friendly solution tailored to the specific demands of mental health care delivery in the UK. By focusing on features such as appointment reminders and secure messaging between professionals, the MHMP seeks to address the limitations of existing general healthcare EHR systems and provide a more targeted approach to managing mental health services.

Through domain-specific research and discussion with healthcare professionals (doctors working in the NHS), we identified a specific need for a messaging system for mental health professionals working in the sector. We therefore included it as an additional feature despite it being extra to the project brief, and through discussion with the client, it was agreed that the MHMP must include a messaging system. This is essential as care for patients in mental health services is often coordinated across hospitals and the community. Due to the importance of patients in the sector taking their medication and the fact that these patients often require extra support to function well in the community, we also decided to implement a notification system for prescriptions, despite this going beyond our project brief and initial discussion with the client. To implement the notification system for appointments, we identified the need to develop a comprehensive calendar system with different views and editing capabilities depending on the type of user.

Our application improves upon existing applications by offering a patient-facing interface, in addition to the MHMP interface used by professionals used in the NHS. Further, there

are specialised legal frameworks within mental health care such as the Mental Health Act (2022) which are not integrated with existing applications.

A challenge for mental health services is that they have to coordinate care for patients across community and hospital settings. It would benefit those working in the sector to have a unified system for coordinating all aspects of a patient's care, bundled within a user-friendly, simple and intuitive design. Healthcare professionals we spoke to when carrying out research discussed the shortcomings of the user interface of existing and their inappropriateness for mental health. Many existing applications have outdated, cluttered designs - working in high pressure situations necessitates a clear user interface for healthcare professionals, and we aim to provide for this need. Please see Appendix 8.1 for a summary of outcomes of discussion with clinicians.

1.5.2 Planned External System Interoperability - The NHS System

Through domain specific research, we identified the need to interface with NHS data in certain instances. For example, the NHS prescription information and GP notes - both are available on the patient-facing NHS app [7]. Throughout this report, we refer to system consisting of NHS databases and information technology as the 'NHS system' - see glossary **Section 2.3**. Through research, we identified that it is possible to access this system and the information contained within through API systems maintained by NHS Digital [8].

1.5.3 Key Features

The MHMP boasts several key features designed to cater to the diverse needs of its users, including patients, pharmacists, therapists / clinical psychologists, psychiatrists, and nurses. These features include:

- Appointment System: Enables appointment scheduling and ensures that all parties are reminded of upcoming appointments, reducing the likelihood of missed sessions.
- Clinical Notes Accessibility: Enables clinical staff to easily access and update notes pertaining to patient care, ensuring a cohesive treatment plan.
- Access to Prescription Information: Allows patients and healthcare professionals to view prescription history and details. Pharmacists can flag prescriptions if there are issues.
- Messaging System for healthcare professional users: Enables smooth communication for staff working across community and hospital care.
- User-Friendly Interface: With a focus on accessibility, MHMP is designed to be intuitive for all users, regardless of their technological proficiency.

1.5.4 Our Report

This report will outline the technical aspects of the design of the MHMP, starting with the requirements, moving into detailed object-oriented modelling utilising the Unified Modelling Language (UML) and visualisation and description of the application mock-up. We provide comprehensive descriptions of the diagrams displayed to facilitate reader understanding.

Chapter 2

Requirements

2.1 Overview

Requirements are important because they provide the basis for all of the development work that follows [9] [10]. We carried out the initial requirements capture process with the client. The list of requirements underwent continuous refinement, in partnership with the client, underscoring the strong connection from the requirements to other components of this report. This iterative process was vital for ensuring that the application remained closely aligned with its goals and demands.

2.2 MoSCoW Requirements

The finalised list of requirements is detailed below. Each requirement has been assigned a priority level according the to MoSCoW method [11] (which involves classing requirements as Must have, Should have, Could have, or Won't have). These requirements have been divided into functional and non-functional requirements. Functional requirements outline the specific behaviours and features of the system, detailing what the system is expected to do. Non-functional requirements specify the system's quality attributes and performance benchmarks. The functional requirements have been categorised by user type for additional clarity and organisation. Through discussion with the client, we agreed that it was desirable for support workers not to have access to our system and instead give access to our system only to nurses.

Functional Requirements:

(a) General:

The first table includes a general requirements for the MHMP system.

ID	Requirement	Priority
FR-G1	The MHMP shall send an account verification email to newly registered users.	Must Have
FR-G2	The MHMP shall display information retrieved from the NHS system, including GP notes and prescriptions.	Must Have
FR-G3	The MHMP shall allow patients, therapists, psychiatrists and secretarial professionals to view appointment information through an in-app calendar.	Must Have
FR-G4	The MHMP shall enforce information access control based on account type.	Must Have
FR-G5	The MHMP shall be able to export the appointments to other calendar applications.	Could Have
FR-G6	The MHMP shall feature a function to allow users to provide feedback.	Won't Have

(b) For Patient Accounts:

The second table includes requirements for the Patient accounts.

ID	Requirement	Priority
FR-P1	The MHMP shall allow patients to edit their email address and password.	Must Have
FR-P2	The MHMP shall allow patients to view their prescription history and details.	Must Have
FR-P3	The MHMP shall allow patients to cancel or request to reschedule an appointment.	Should Have
FR-P4	The MHMP shall send in-app notifications for prescriptions and appointments.	Should Have
FR-P5	The MHMP shall allow patients to request repeat prescriptions.	Should Have
FR-P6	The MHMP shall feature educational resources on mental health for patients.	Won't Have

(c) For Healthcare Professional Accounts:

The third table includes requirements for the Healthcare Professional accounts.

ID	Requirement	Priority
FR-H1	The MHMP shall allow psychiatrists to create and update prescriptions.	Must Have
FR-H2	The MHMP shall have authorship and viewing ability for clinical notes. (see Table 2.2)	Must Have
FR-H3	The MHMP shall allow the healthcare professionals to search, filter, and view patient information.	Must Have
FR-H4	The MHMP shall provide a messaging system for non-patient users to communicate.	Must Have
FR-H5	The MHMP shall allow hospital pharmacists to flag concerns about prescriptions.	Should Have
FR-H6	The MHMP shall allow psychiatrists to receive flagged prescription notifications.	Should Have
FR-H7	The MHMP shall send in-app notifications for appointments.	Should Have
FR-H8	The MHMP shall provide healthcare professionals advanced analytics on patient progress.	Could Have
FR-H9	The MHMP shall allow healthcare professionals to generate patient reports.	Won't Have

(d) For Secretarial Professional Accounts:

The third table includes requirements for the Secretarial Professional accounts.

ID	Requirement	Priority
FR-S1	The MHMP shall allow secretarial professionals to manage appointments.	Must Have
FR-S2	The MHMP shall allow secretarial professionals to receive rescheduling requests.	Must Have
FR-S3	The MHMP shall allow secretarial professionals to edit a patient's details, including their mental health act status and clinical status.	Must Have
FR-S4	The MHMP shall allow secretarial professionals to create and manage non-admin accounts.	Must Have

(e) For System Administrator Accounts:

The third table includes requirements for the System Administrator accounts.

ID	Requirement	Priority
FR-A1	The MHMP shall allow system administrators to create and manage all user accounts.	Must Have
FR-A2	The MHMP shall allow system administrators to review system logs.	Must Have

Non-Functional Requirements:

This table includes the non-functional requirements for all site users, registered or unregistered. It represents the purposes and goals of the site.

ID	Requirement	Priority
NFR-1	The MHMP shall be available for 99.999% of the time, with scheduled downtime for maintenance.	Must Have
NFR-2	The MHMP shall provide responsive UI for any screen sizes.	Must Have
NFR-3	The MHMP shall support Chromium-based, Safari, and Firefox browsers.	Must Have
NFR-4	The MHMP shall be available as a downloadable app on both Google Android and iOS.	Must Have
NFR-5	The MHMP shall encrypt the storage of sensitive data.	Must Have
NFR-6	The MHMP shall regularly backup data daily as a recovery mechanism.	Must Have
NFR-7	The MHMP shall comply with The Health Insurance Portability and Accountability Act of 1996 (HIPAA) and General Data Protection Regulation (GDPR).	Must Have
NFR-8	The MHMP shall process data in a secure way with encryption.	Must Have
NFR-9	The MHMP shall interface with the NHS system to retrieve data.	Must Have
NFR-10	The MHMP shall interface with the NHS system to update data and pass user requests.	Must Have
NFR-11	The MHMP shall be accessible to all users and comply with Web Content Accessibility Guidelines (WCAG).	Should Have
NFR-12	The MHMP shall process user inputs / requests on average within 2 seconds.	Should Have
NFR-13	The MHMP shall scale to large user loads without a noticeable loss in performance.	Should Have

2.3 Glossary

Item	Definition
Healthcare Professional	Clinical healthcare workers within the NHS who use the MHMP system.
Psychiatrist	Healthcare professional who has completed a medical degree and is undergoing training / has completed postgraduate psychiatry training. They are specialists in the prescription of medication for mental health conditions.
Therapist	Synonymous with clinical psychologist for the purposes of the MHMP. A healthcare professional who evaluates and assesses patients' mental health. They offer talking therapy sessions with patients and help formulate treatment plans.

Item	Definition
Hospital Pharmacist	A specialised psychiatric pharmacist based in a hospital who reviews medication details, and ensures a safe and effective medication treatment for patients. Responsible for flagging medications in our system.
Pharmacist	Synonymous with hospital pharmacist for the purposes of the MHMP.
Nurse	Mental health nurses who have completed formal training in mental health nursing.
Secretarial Professionals	Medical secretaries embedded throughout the NHS mental health system.
System Administrator (Admin)	Information technology expert who has privileges to facilitate the maintenance of the system and has received formal training in data protection and associated regulations.
Patient	An individual receiving mental health services from mental health trusts using the MHMP platform.
Patient Information	The complete set of details held on the MHMP system about a particular patient including patient details, prescription information and clinical notes
Patient Details	Includes name, date of birth, NHS number, clinical status, mental health act status.
Mental Health Act Status	The status of detention under the Mental Health Act (2022). This Act gives psychiatrists and nurses the ability to detain patients within a psychiatric hospital for protection of themselves or others. It is important for care of patients with mental health problems.
Clinical Status	List values pertaining to the current status of the patient including: • Hospitalised. • Intensive community support (for example, receiving regular care by psychiatrists and other healthcare professionals whilst living at home). • General community support (including regular talking therapy, continuous check-ups, and/or prescription of medication). • Not receiving support.
Clinical Notes	Four types of clinical notes available: • GP notes. • Psychiatric notes, which include: – Discharge letters. – Communication letters. – Notes from psychiatric assessments. – Procedures (such as blood tests, electrocardiogram recordings, electroconvulsive therapy sessions). • Therapist notes. • General observations (for nurses to record observations).

Item	Definition
GP Notes	Notes collected from a patient's general practitioner (GP), available on central NHS databases in the UK.
Psychiatric Notes	Notes collected during treatment by psychiatrists. This will also contain details of procedures in psychiatry carried out such as electrocardiograms (ECG), repetitive transcranial magnetic stimulation (rTMS), blood tests, and electro-convulsive therapy (ECT).
Therapist Notes	Notes collected during treatment by (talking / psychological) therapists / clinical psychologists.
Responsive UI	An interface that dynamically resizes depending on the size of the viewing window without restricting viewing.
NHS System	The collection of databases on which prescription history and GP notes are held.

2.4 Read and Write Privileges

We have been careful to establish consistent read and write privileges for different types of healthcare professional across patient information (including patient details, prescription information and clinical notes) and appointments. Please see **Table 2.2** below. These read write privileges correspond to the following requirements: FR-G2, FR-G4, FR-H1, FR-H2, FR-H3, FR-H5, FR-S1, FR-S3.

Type of Information			**Psychiatrist**	**Nurse**	**Therapist**	**Hospital Pharmacist**	**Secretarial Professional**
Patient Information		Patient Details	R	R	R	R	W
		Prescription	W	R	R	R	R
	Clinical Notes	GP Notes	R	R	-	R	R
		Psychiatric Notes	W	W	R	R	-
		Therapist Notes	R	-	W	-	-
		General Observations	R	W	R	R	R
		Appointments	W	-	W	-	W

Table 2.2: Information Read and Write Privileges by User Type

2.5 Domain Model

The UML domain model shows aggregation and generalisation relationships (has-a and is-a relationships) between the domain classes [12]. Consisting of entities and entity relationships, the domain model provides a common language for all parties and thus improves the clarity of the requirements and the system's design. The first-pass domain model was developed by identifying objects directly from the requirements list. To refine this model, the ICONIX approach: *continuous improvement via ongoing iteration and refinement*, was leveraged to complete the second-pass domain model [12], which is displayed in **Figure 2.5.0.1** below.

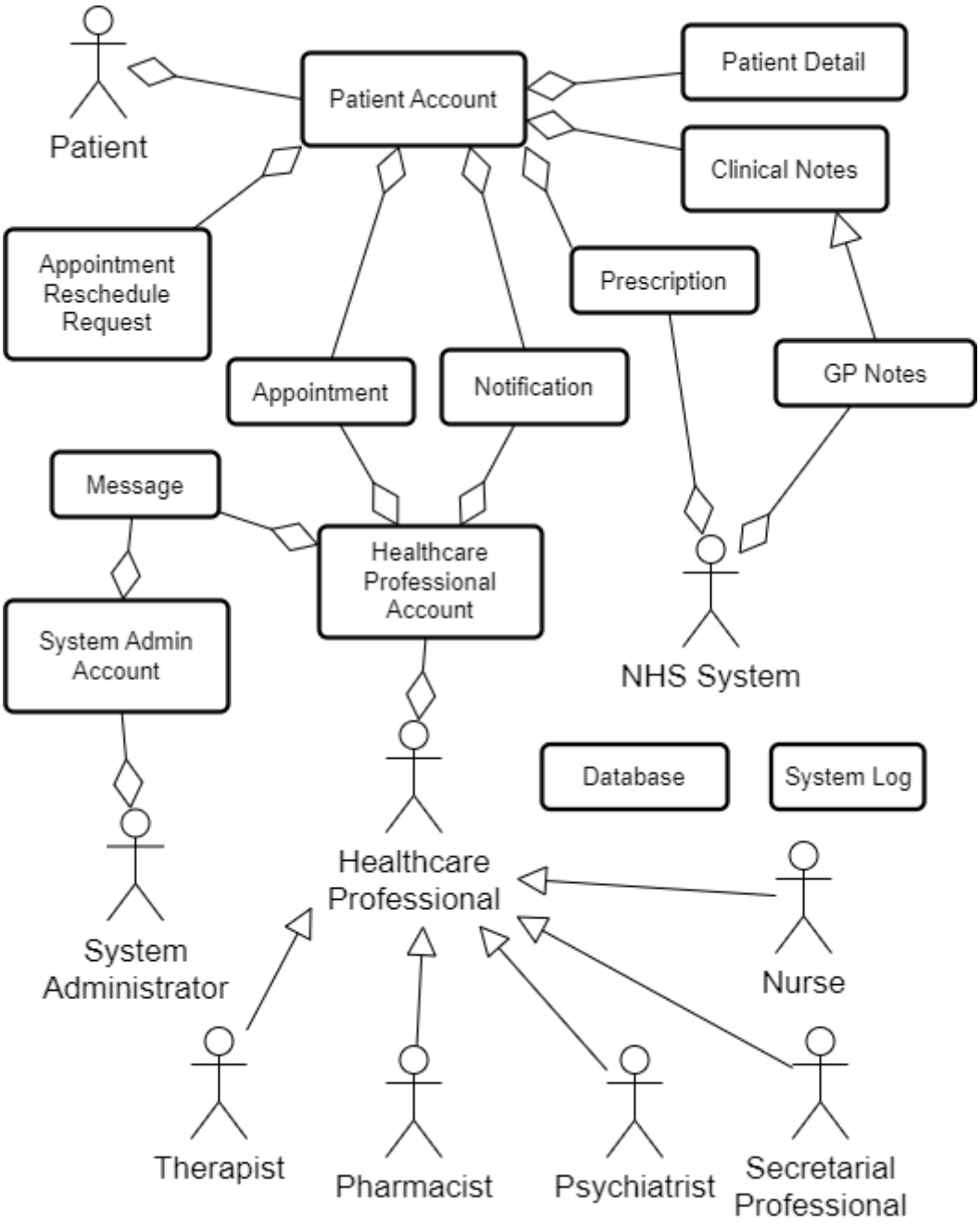


Figure 2.5.0.1: Domain Model

Chapter 3

Use Cases

3.1 Use Case Overview

A use case is a description of the interactions and responsibilities of a system with external agents, or actors [13]. Use cases are excellent for building test cases and procedures because they precisely capture a user's requirements and success criteria [14]. The use cases drawn up are strongly linked to the requirements detailed in the previous chapter. This chapter presents a comprehensive list of all the use cases identified, a requirements traceability matrix to map these use cases to their corresponding requirements, a use case diagram that visualises the system's interactions with its actors, and detailed use case specifications that describe the sequence of actions for each use case.

3.2 Use Case Listing

Use cases for the MHMP were developed by reviewing the requirements and the domain model, defining the system boundaries, and identifying the actors. Detailed scenarios underwent multiple iterations to present comprehensive flows that capture the requirements listed. To facilitate the use case analysis, actors were categorised into roles based on their interaction with the system. This categorisation is displayed in **Table 3.1** and the final list of use cases is displayed in **Table 3.2** below.

Actor	Role			
	User	**Healthcare Professional**	**Clinical Note Manager**	**Appointment Manager**
Psychiatrist	✓	✓	✓	✓
Therapist	✓	✓	✓	✓
Secretarial Professional	✓	✓	×	✓
Nurse	✓	✓	✓	×
Hospital Pharmacist	✓	✓	×	×
Patient	✓	×	×	×
System Administrator	×	×	×	×
NHS System	×	×	×	×

Table 3.1: Actors and Groupings

ID	Use Case Name	Actor(s)
UC1	ManageAdminAccounts	System Administrator
UC2	ManageNonAdminAccounts	System Administrator, Secretarial Professional
UC3	UpdateAccountDetails	User
UC4	UpdatePatientDetails	Secretarial Professional
UC5	ViewPatientDetails	User
UC6	ViewClinicalNotes	Healthcare Professional
UC7	ManageClinicalNotes	Clinical Note Manager
UC8	ManageAppointment	Appointment Manager
UC9	ViewAppointment	Appointment Manager, Patient
UC10	CancelAppointment	Patient
UC11	RequestToRescheduleAppointment	Patient
UC12	ProcessAppointmentRescheduleRequest	Secretarial Professional
UC13	ReceiveAppointmentNotification	Patient, Psychiatrist, Therapist
UC14	ManagePrescription	Psychiatrist
UC15	ReceivePrescriptionNotification	Patient
UC16	ViewPrescription	User
UC17	FlagPrescriptionForReview	Hospital Pharmacist
UC18	ReceiveFlaggedPrescriptionNotification	Psychiatrist
UC19	RequestRepeatPrescription	Patient
UC20	SynchoniseData	NHS System
UC21	SendMessages	System Administrator, Healthcare Professional
UC22	ReviewSystemLogs	System Administrator

Table 3.2: Use Case Listing.

3.3 Requirements Traceability Matrix

The Requirements Traceability Matrix maps each functional requirement to its relevant use case(s), and conversely, ensuring comprehensive coverage of the requirements through the use cases. This matrix helps to trace the requirements laid out earlier, verifying that they have been addressed in the specified use case(s). It is noted that certain low-priority requirements without a use case have been greyed out. Non-functional requirements do not consist of use cases and been omitted from the matrix. The matrix is displayed in Table 3.3 below.

	UC																				
Req	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
FR-G1	X	X																				
FR-G2						X										X				X		
FR-G3									X													
FR-G4					X	X										X						
FR-G5	Could Have Requirement																					
FR-G6	Won't Have Requirement																					
FR-P1			X																			
FR-P2																X						
FR-P3										X	X											
FR-P4													X		X							
FR-P5																			X			
FR-P6	Won't Have Requirement																					
FR-P7	Won't Have Requirement																					
FR-H1														X								
FR-H2						X	X															
FR-H3					X																	
FR-H4																					X	
FR-H5																	X					
FR-H6																		X				
FR-H7													X									
FR-H8	Won't Have Requirement																					
FR-H9	Won't Have Requirement																					
FR-S1								X														
FR-S2												X										
FR-S3				X																		
FR-S4		X																				
FR-A1	X	X																				
FR-A2																						X

Table 3.3: Requirements Traceability Matrix

3.4 Use Case Diagram

The use case diagram includes all use cases generated, representing and showing the interactions between actors and the system. The <<include>> stereotype is used to include a use case's behaviour into the flow of another use case, and indicate that a use case is mandatory for another to take place. Actor generalisation is also shown by arrows between actors, for example, a therapist or a nurse is a clinical note manager and inherits its related use case(s). The use case diagram is displayed in Figure 3.4.0.1 below:

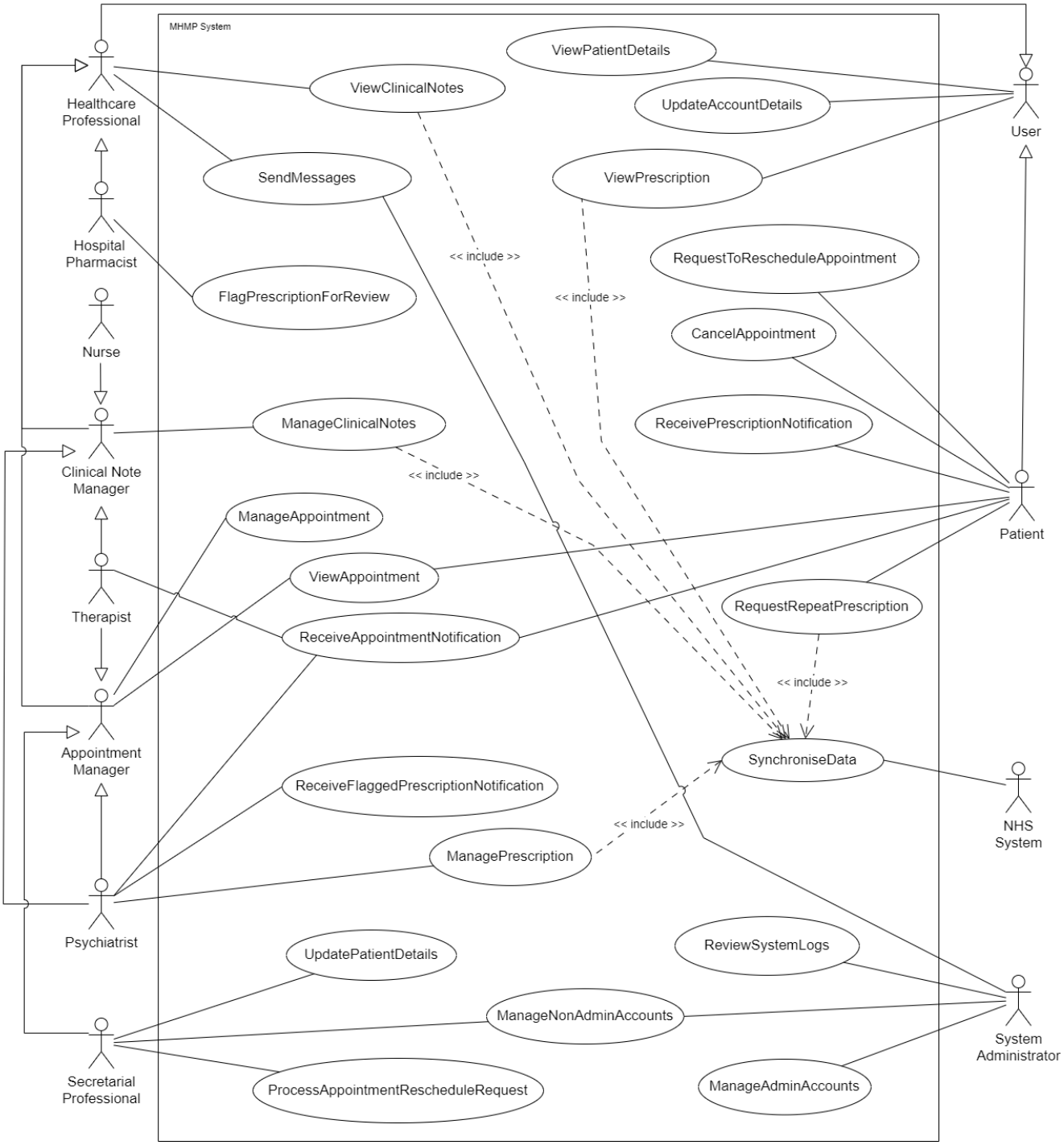


Figure 3.4.0.1: MHMP Use Case Diagram

3.5 Use Case Specifications

The use case specifications elaborate each use case by providing detailed descriptions for not only the main flow, but also any alternative flows that might arise. These specifications capture the interactions between the system and the actors, which include system administrators, secretarial professionals, patients, psychiatrists, hospital pharmacists, therapists, nurses and the NHS system. By detailing the expected flows, a greater understanding of how the system will operate across different scenarios can be achieved.

Name:	ManageAdminAccounts
ID:	UC1
Brief Description	System Administrators create or update admin User accounts within the MHMP System.
Primary Actor(s):	System Administrators.
Secondary Actor(s):	None.
Pre-conditions:	1. The System Administrator must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the System Administrator navigates to the 'Account Management' page within the MHMP. 2. If they wish to create a new admin User account, then 2.1 They select the option to create a new account. 2.2 They fill out the required information for the new admin User. 3. If they wish to update an existing admin User account, then 3.1 They search for the admin User account by using the search functionality. 3.2 They then select the admin User account to update. 3.3 They edit the User account's information. 3.4 They save and confirm the changes. 4. The MHMP System creates or updates the user record in the MHMP database. 5. They exit the 'Account Management' page or log out.
Post-conditions:	1. The user record is created or updated in the MHMP database.
Alternative Flows:	1. UpdateFailed

Name:	ManageAdminAccounts: UpdateFailed
ID:	UC1.1
Brief Description	The MHMP System failed to update the user record in the database.
Primary Actor(s):	System Administrators.
Secondary Actor(s):	None.
Pre-conditions:	1. The System Administrator has created or updated an admin User account.
Alternative Flow:	1. The alternative flow starts during step 4 of the main flow. 2. The MHMP System fails to update the user record in the database. 3. The MHMP System notifies them that the update has failed. 4. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	ManageNonAdminAccounts
ID:	UC2
Brief Description	System Administrators and Secretarial Professionals manage non-admin User accounts within the MHMP System.

Primary Actor(s):	System Administrators, Secretarial Professionals.
Secondary Actor(s):	None.
Pre-conditions:	1. The System Administrator or Secretarial Professional must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the System Administrator or Secretarial Professional navigates to the 'Account Management' page within the MHMP. 2. If they wish to create a new User account, then 2.1 They select the option to create a new account. 2.2 They fill out the required information for the new User. 3. If they wish to update an existing User account, then 3.1 They search for the User account by using the search functionality. 3.2 If they are a Secretarial Professional, then 3.2.1 Only non-admin User accounts will be displayed. 3.3 They then select the User account to update. 3.4 They edit the User account's information 3.5 They save and confirm the changes. 4. The MHMP System creates or updates the user record in the MHMP database. 5. They exit the 'Account Management' page or log out.
Post-conditions:	1. The user record is created or updated in the MHMP database.
Alternative Flows:	1. UpdateFailed

Name:	ManageNonAdminAccounts: UpdateFailed
ID:	UC2.1
Brief Description	The MHMP System failed to update the user record in the database.
Primary Actor(s):	System Administrators, Secretarial Professionals.
Secondary Actor(s):	None.
Pre-conditions:	1. The System Administrator or Secretarial Professional has created or updated a User account.
Alternative Flow:	1. The alternative flow starts during step 4 of the main flow. 2. The MHMP System fails to update the user record in the database. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	UpdateAccountDetails
ID:	UC3
Brief Description	Users update their account details.
Primary Actor(s):	Users (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals, Patients).
Secondary Actor(s):	None.
Pre-conditions:	1. The User must be logged onto the MHMP System.

Main Flow:	1. The use case starts when the User navigates to the 'Personal Details' page on the MHMP. 2. They edit their email address or password. 3. They select the submit option. 4. While the MHMP System detects a validation failure, including invalid email address or mismatch new passwords 4.1 The MHMP System alerts them to check the relevant input field(s). 4.2 They update the field(s) and select the submit option. 5. The MHMP System requests for their current password to perform the update. 6. They enter their current password to authorise the change. 7. While they have entered a wrong current password 7.1 The MHMP System rejects the request and they may try again. 8. The MHMP System updates the user record in the MHMP database. 9. They exit the 'Personal Details' page or log out.
Post-conditions:	1. The user record is updated in the MHMP database.
Alternative Flows:	1. UpdateFailed

Name:	UpdateAccountDetails: UpdateFailed
ID:	UC3.1
Brief Description	The MHMP System failed to update the user record in the database.
Primary Actor(s):	Users (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals, Patients).
Secondary Actor(s):	None.
Pre-conditions:	1. The User has submitted their new email address or password.
Alternative Flow:	1. The alternative flow starts during step 8 of the main flow. 2. The MHMP System fails to update the user record in the database. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	UpdatePatientDetails
ID:	UC4
Brief Description	Secretarial Professionals update Patient details, including mental health act status, clinical status and comments.
Primary Actor(s):	Secretarial Professionals.
Secondary Actor(s):	None.
Pre-conditions:	1. The Secretarial Professional must be logged onto the MHMP System.

Main Flow:	1. The use case starts when the Secretarial Professional navigates to the 'Select Patient' page on the MHMP. 2. They search for a Patient by name or NHS number. 3. While the search results do not return the Patient 3.1 They may try again with different search criteria. 4. They select a Patient from the list of Patients matching the search criteria. 5. The MHMP System retrieves and displays the Patient's details. 6. They review the information and select the edit patient details option. 7. They update the necessary information, which may include mental health act status, clinical status and comments. 8. They save the updated information. 9. The MHMP System updates the patient record in the MHMP database. 10. They exit the 'Select Patient' page or log out.
Post-conditions:	1. The patient record is updated in the MHMP database.
Alternative Flows:	1. UpdateFailed

Name:	UpdatePatientDetails: UpdateFailed
ID:	UC4.1
Brief Description	The MHMP System failed to update the patient record in the database.
Primary Actor(s):	Secretarial Professionals.
Secondary Actor(s):	None.
Pre-conditions:	1. The Secretarial Professional has edited and saved a Patient's details.
Alternative Flow:	1. The alternative flow starts during step 9 of the main flow. 2. The MHMP System fails to update the patient record in the database. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	ViewPatientDetails
ID:	UC5
Brief Description	Users view Patients' details, including mental health act status, clinical status and comments.
Primary Actor(s):	Users (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals, Patients).
Secondary Actor(s):	None.
Pre-conditions:	1. The User must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the User navigates to the 'Select Patient' page on the MHMP. 2. They search for a Patient by name or NHS number. 3. While the search results do not return the Patient 3.1 They may try again with different search criteria. 4. They select a Patient from the list of Patients matching the search criteria. 5. The MHMP System retrieves and displays the Patient's details. 6. They review the information and may perform further actions. 7. They exit the 'Select Patient' page or log out.
Post-conditions:	None.
Alternative Flows:	None.

Name:	ViewClinicalNotes
ID:	UC6
Brief Description	Authorised Healthcare Professionals view relevant clinical notes on the MHMP System.
Primary Actor(s):	Healthcare Professionals (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals).
Secondary Actor(s):	NHS System.
Pre-conditions:	1. The Healthcare Professional must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the Healthcare Professional navigates to the 'Select Patient' page on the MHMP. 2. They search for a Patient by name or NHS number. 3. While the search results do not return the Patient 3.1 They may try again with different search criteria. 4. They select a Patient from the list of Patients matching the search criteria. 5. The MHMP System displays the tabs for the types of clinical notes based on their role. 6. They select the type of clinical notes to view. 7. If the type of clinical note selected is GP note, then 7.1 The MHMP System retrieves GP notes from the NHS System. 8. Else 8.1 The MHMP System retrieves the clinical notes from its database. 9. The MHMP System displays the clinical notes. 10. They exit the 'Select Patient' page or log out.
Post-conditions:	None.
Alternative Flows:	None.

Name:	ManageClinicalNotes
ID:	UC7
Brief Description	Clinical Note Managers create or edit a relevant clinical note on the MHMP System.
Primary Actor(s):	Clinical Note Managers (Psychiatrists, Therapists, Nurses).
Secondary Actor(s):	NHS System.
Pre-conditions:	1. The Clinical Note Manager must be logged onto the MHMP System.

Main Flow:	1. The use case starts when the Clinical Note Manager navigates to the 'Select Patient' page on the MHMP. 2. They search for a Patient by name or NHS number. 3. While the search results do not return the Patient 3.1 They may try again with different search criteria. 4. They select a Patient from the list of Patients matching the search criteria. 5. They select the tab of the type of clinical notes to view. 6. The MHMP System retrieves and displays the Patient's clinical notes. 7. If they wish to create a new clinical note, then 7.1 They select the option to create a new clinical note. 7.2 They fill out all the relevant information. 7.3 They submit the clinical note. 8. If they wish to edit a clinical note, then 8.1 They select an existing clinical note to edit. 8.2 They select the option to edit the clinical note. 8.3 They perform any relevant edits on the clinical note. 8.4 They submit and save the changes made to the clinical note. 9. The MHMP System creates or updates the clinical notes record in the MHMP database. 10. They exit the 'Select Patient' page or log out.
Post-conditions:	1. The clinical notes record is created or updated in the MHMP database.
Alternative Flows:	None.

Name:	ManageAppointment
ID:	UC8
Brief Description	Appointment Managers create or edit appointments.
Primary Actor(s):	Appointment Managers (Psychiatrists, Therapists, Secretarial Professionals).
Secondary Actor(s):	Patient, Psychiatrists, Therapists.
Pre-conditions:	1. The Appointment Manager must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the Appointment Manager navigates to the 'Appointment Management' page of the MHMP. 2. If they wish to create a new appointment, then 2.1 They select the option to create a new appointment. 2.2 They fill out all the necessary appointment information. 2.3 They submit to create the appointment. 3. If they wish to edit an existing appointment, then 3.1 They select an appointment to edit from the list of appointments through the in-app calendar. 3.2 They edit any appointment information as required. 3.3 They submit to update the appointment. 4. The MHMP System creates or updates the appointment record in the MHMP database. 5. The MHMP System sends notifications to the Patient and the Appointment Manager(s) containing information of the appointment. 6. They exit the 'Appointment Management' page or log out.
Post-conditions:	1. The appointment record is created or updated in the MHMP database.
Alternative Flows:	1. UpdateFailed

Name:	ManageAppointment: UpdateFailed

ID:	UC8.1
Brief Description	The MHMP System failed to update the appointment record in the database.
Primary Actor(s):	Appointment Managers (Psychiatrists, Therapists, Secretarial Professionals).
Secondary Actor(s):	None.
Pre-conditions:	1. The Appointment Manager has created or edited an appointment.
Alternative Flow:	1. The alternative flow starts during step 4 of the main flow. 2. The MHMP System fails to update the appointment record in the database. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	ViewAppointment
ID:	UC9
Brief Description	Appointment Managers and Patients view upcoming appointments on the MHMP.
Primary Actor(s):	Appointment Managers (Psychiatrists, Therapists, Secretarial Professionals), Patients.
Secondary Actor(s):	None.
Pre-conditions:	1. The Appointment Manager or Patient must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the Appointment Manager navigates to the 'Appointment Management' page or the Patient navigates to the 'Your Appointments' page of the MHMP. 2. The MHMP System displays a list of all their upcoming appointments through the in-app calendar. 3. They view an appointment's details, such as the appointment's location, time, and name of the Healthcare Professional. 4. They exit the 'Appointment Management' page, 'Your Appointments' page, or log out.
Post-conditions:	None.
Alternative Flows:	1. NotFound

Name:	ViewAppointment: NotFound
ID:	UC9.1
Brief Description	The Appointment Manager or Patient failed to retrieve their appointment.
Primary Actor(s):	Appointment Managers (Psychiatrists, Therapists, Secretarial Professionals), Patients.
Secondary Actor(s):	None.
Pre-conditions:	1. The Appointment Manager or Patient must be logged onto the MHMP System.
Alternative Flow:	1. The alternative flow starts after step 2 of the main flow. 2. They do not find a specific appointment. 3. They may check their notifications for a cancellation or rescheduling notice. 4. They may contact a relevant Secretarial Professional to create an appointment.
Post-conditions:	None.

Name:	CancelAppointment
ID:	UC10
Brief Description	Patients cancel their appointments.
Primary Actor(s):	Patients.
Secondary Actor(s):	Appointment Managers (Psychiatrists, Therapists, Secretarial Professionals).
Pre-conditions:	1. The Patient has an upcoming appointment on the MHMP System. 2. The Patient must be logged onto the MHMP System. 3. The cancellation attempt is being made within the pre-defined time window.
Main Flow:	1. The use case starts when the Patient navigates to the 'Appointment Management' page of the MHMP. 2. The MHMP System displays a list of all their upcoming appointments through the in-app calendar. 3. They select the option to cancel an appointment. 4. They select to cancel the appointment. 5. The MHMP System updates the appointment record in the MHMP database. 6. The MHMP System notifies the relevant Appointment Manager. 7. They exit the 'Appointment Management' page or log out.
Post-conditions:	1. The appointment record is updated in the MHMP database. 2. The MHMP System notifies the Appointment Manager of the appointment cancellation.
Alternative Flows:	1. UpdateFailed

Name:	CancelAppointment: UpdateFailed
ID:	UC10.1
Brief Description	The MHMP System failed to update the appointment record in the database.
Primary Actor(s):	Patients.
Secondary Actor(s):	None.
Pre-conditions:	1. The Patient has selected to cancel an appointment.
Alternative Flow:	1. The alternative flow starts after step 4 of the main flow. 2. The MHMP System fails to update the appointment record in the database. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	RequestToRescheduleAppointment
ID:	UC11
Brief Description	Patients submit requests to reschedule their appointments.
Primary Actor(s):	Patients.
Secondary Actor(s):	None
Pre-conditions:	1. The Patient has an upcoming appointment on the MHMP System. 2. The Patient must be logged onto the MHMP System. 3. The reschedule request is being made within the pre-defined time window.

Main Flow:	1. The use case starts when the Patient navigates to the 'Appointment Management' page of the MHMP. 2. The MHMP System displays a list of all their upcoming appointments through the in-app calendar. 3. They select the option to reschedule an appointment. 4. They may enter their preferred reschedule timing. 5. They submit their request to reschedule the appointment. 6. The MHMP System creates the appointment reschedule request record in the MHMP database. 7. They exit the 'Appointment Management' page or log out.
Post-conditions:	1. The appointment reschedule request record is created in the MHMP database. 2. The Secretarial Professional is notified of the request to reschedule the appointment.
Alternative Flows:	1. UpdateFailed

Name:	RequestToRescheduleAppointment: UpdateFailed
ID:	UC11.1
Brief Description	The MHMP System failed to process the request.
Primary Actor(s):	Patients.
Secondary Actor(s):	None.
Pre-conditions:	1. The Patient has submitted a request to reschedule an appointment.
Alternative Flow:	1. The alternative flow starts after step 5 of the main flow. 2. The MHMP System fails to process the request. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	ProcessAppointmentRescheduleRequest
ID:	UC12
Brief Description	Secretarial Professionals receive and process requests to reschedule appointments.
Primary Actor(s):	Secretarial Professionals.
Secondary Actor(s):	Patients, Psychiatrists, Therapists.
Pre-conditions:	1. The Patient has submitted a request to reschedule an appointment. 2. The Secretarial Professional must be logged onto the MHMP System.

Main Flow:	1. The Secretarial Professional logs into the MHMP System and navigates to the 'Appointment Management' page. 2. They select the option to view the reschedule requests. 3. They select the appointment to process the reschedule. 4. They check the calendar of the Healthcare Professional involved. 5. If the Patient has suggested timings and the Healthcare Professional is also available, then 5.1 They select the time that accommodates both parties. 6. Else 6.1 They select the Healthcare Professional's next available time. 7. They update the appointment with the new time. 8. The MHMP System updates the appointment record in the MHMP database. 9. The MHMP System updates the appointment reschedule request record in the MHMP database. 10. The MHMP System sends a notification to the Patient and the Healthcare Professional informing them of the rescheduled appointment time. 11. They exit the 'Appointment Management' page or log out.
Post-conditions:	1. The appointment record and the appointment reschedule request record are updated in the MHMP database. 2. The Patient is notified of the rescheduled appointment time.
Alternative Flows:	1. UpdateFailed

Name:	ProcessAppointmentRescheduleRequest: UpdateFailed
ID:	UC12.1
Brief Description	The MHMP System failed to update the appointment reschedule request database.
Primary Actor(s):	Secretarial Professionals.
Secondary Actor(s):	None.
Pre-conditions:	1. The Secretarial Professional has rescheduled an appointment.
Alternative Flow:	1. The alternative flow starts after step 8 of the main flow. 2. The MHMP System fails to update the appointment database. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	ReceiveAppointmentNotification
ID:	UC13
Brief Description	Patients, Psychiatrists, and Therapists receive a notification when they have an appointment that is created or updated.
Primary Actor(s):	Patients, Psychiatrists, Therapists.
Secondary Actor(s):	None.
Pre-conditions:	An appointment involving the primary actors has been created or updated.
Main Flow:	1. The use case starts with the MHMP generating a notification containing appointment details. 2. They see the notification present in the notifications tab featuring the appointment details.
Post-conditions:	1. None.
Alternative Flows:	1. NotificationFailure

Name:	ReceiveAppointmentNotification: NotificationFailure
ID:	UC13.1
Brief Description	The prescription notification failed to be received by the User.
Primary Actor(s):	Patients, Psychiatrists, Therapists.
Secondary Actor(s):	None.
Pre-conditions:	1. The MHMP has generated and sent out the notification to the Patient. 2. The patient receives the notification and can view the details.
Alternative Flow:	1. The alternative flow starts during step 1 of the main flow. 2. The notification fails to be sent due to errors such as system error or connection error. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	ManagePrescription
ID:	UC14
Brief Description	Psychiatrists create or update existing prescriptions for Patients.
Primary Actor(s):	Psychiatrists.
Secondary Actor(s):	Patients, NHS System.
Pre-conditions:	1. The Psychiatrist must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the Psychiatrist navigates to the 'Select Patient' page within the MHMP. 2. They search for a Patient by name or NHS number. 3. While the search results do not return the Patient 3.1 They may try again with different search criteria. 4. They select a Patient from the list of Patients matching the search criteria. 5. They select the prescription tab. 6. If they wish to create a new prescription, then 6.1 They select the option to create a new prescription. 6.2 They fill out the relevant prescription information, including medication name, dosage, frequency, duration, and any special instructions. 7. If they wish to edit an existing prescription, then 7.1 They select an existing prescription. 7.2 They select the option to edit the prescription. 7.3 They edit any necessary prescription information. 8. They submit the prescription. 9. The MHMP System sends the new or updated prescription details to be updated in the NHS database. 10. The MHMP System creates or updates the prescription record in the MHMP database. 11. They exit the 'Select Patient' page or log out.
Post-conditions:	1. The MHMP database and NHS database is updated accordingly.
Alternative Flows:	1. Cancel 2. IncompleteInput

Name:	ManagePrescription: Cancel
ID:	UC14.1

Brief Description	The Psychiatrist cancels creating or editing a prescription.
Primary Actor(s):	Psychiatrists.
Secondary Actor(s):	None.
Pre-conditions:	1. The Psychiatrist has selected to create or edit a prescription.
Alternative Flow:	1. The alternative flow starts after steps 6 or 7 of the main flow. 2. The Psychiatrist does not submit the prescription and exits the prescription form page.
Post-conditions:	None.

Name:	ManagePrescription: IncompleteInput
ID:	UC14.2
Brief Description	The MHMP System informs the Psychiatrist that the input information is incomplete.
Primary Actor(s):	Psychiatrists.
Secondary Actor(s):	None.
Pre-conditions:	1. The Psychiatrist has selected to create or edit a prescription.
Alternative Flow:	1. The alternative flow starts after step 8 of the main flow. 2. The Psychiatrist does not fill out all required information. 3. The MHMP System informs them of the missing information.
Post-conditions:	None.

Name:	ReceivePrescriptionNotification
ID:	UC15
Brief Description	Patients receive notifications for prescriptions created for them.
Primary Actor(s):	Patients.
Secondary Actor(s):	None.
Pre-conditions:	A prescription has been created for the patient.
Main Flow:	1. The use case starts with the MHMP generating a notification containing prescription details. 2. They receive the notification and can view via their notification tab.
Post-conditions:	1. None.
Alternative Flows:	1. NotificationFailure

Name:	ReceivePrescriptionNotification: NotificationFailure
ID:	UC15.1
Brief Description	The prescription notification failed to be received by the Patient.
Primary Actor(s):	Patient.
Secondary Actor(s):	None.

Pre-conditions:	1. The Psychiatrist has created or updated a Patient's prescription.
Alternative Flow:	1. The alternative flow starts during step 1 of the main flow. 2. The notification fails to be sent due to errors such as system error or connection error. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	ViewPrescription
ID:	UC16
Brief Description	Users view Patients' prescriptions.
Primary Actor(s):	Users (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals, Patients).
Secondary Actor(s):	NHS System.
Pre-conditions:	1. The User must be logged onto the MHMP System.
Main Flow:	1. If the User is a Patient, then 1.1 The use case start when the User navigates to the 'Your Prescriptions' page within the MHMP. 2. Else 2.1 The use case starts when the User views the prescriptions page for a particular patient within the MHMP. 3. The MHMP System retrieves the Patient's prescription details from the NHS System and displays them. 4. They review the information. 5. They exit the 'Your Prescriptions' page, 'Select Patient' page, or log out.
Post-conditions:	None.
Alternative Flows:	None.

Name:	FlagPrescriptionForReview
ID:	UC17
Brief Description	Hospital Pharmacists flag prescriptions to be reviewed by the prescribing Psychiatrist.
Primary Actor(s):	Hospital Pharmacists.
Secondary Actor(s):	None.
Pre-conditions:	1. The Pharmacist must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the Pharmacist navigates to the prescription tab for a patient. 2. They select the prescription to be flagged for review. 3. The MHMP System updates the prescription record in the MHMP database. 4. They exit the 'Select Patient' page or log out.
Post-conditions:	1. The prescription record is updated in the MHMP database.
Alternative Flows:	None.

Name:	ReceiveFlaggedPrescriptionNotification
ID:	UC18

Brief Description	Psychiatrists receive notifications to review a prescription.
Primary Actor(s):	Psychiatrists.
Secondary Actor(s):	None.
Pre-conditions:	1. The Pharmacist has flagged a prescription to be reviewed. 2. The Psychiatrist must be logged onto the MHMP system.
Main Flow:	1. The use case starts with the MHMP generating a notification regarding the flagged prescription. 2. The Psychiatrist can then view the notification in the notifications tab.
Post-conditions:	1. None.
Alternative Flows:	1. FurtherClarification

Name:	ReceiveFlaggedPrescriptionNotification: FurtherClarification
ID:	UC18.1
Brief Description	The Psychiatrist requires further clarification from the Pharmacist.
Primary Actor(s):	Psychiatrists.
Secondary Actor(s):	None.
Pre-conditions:	1. The Psychiatrist reviewed the prescription flagged by the Pharmacist.
Alternative Flow:	1. The alternative flow starts after step 2 of the main flow. 2. The Psychiatrist contacts the Pharmacist through the MHMP's messaging feature or any other preferred method. 3. They request for additional information regarding the flagged prescription.
Post-conditions:	1. The message has been sent.

Name:	RequestRepeatPrescription
ID:	UC19
Brief Description	Patients submit requests for repeat prescriptions.
Primary Actor(s):	Patients.
Secondary Actor(s):	NHS System.
Pre-conditions:	1. The Patient must be logged onto the MHMP System. 2. The prescription must be issued and repeatable.
Main Flow:	1. The use case starts when the Patient navigates to the 'Your Prescriptions' page within the MHMP. 2. They select the prescription they wish to have refilled. 3. They select the option to request for a repeat prescription. 4. The MHMP System sends the request to the NHS database. 5. Once approved or denied, the NHS System sends the request outcome back to the MHMP System. 6. The MHMP System updates the prescription record in the MHMP database.
Post-conditions:	1. The prescription record is updated in the MHMP database. 2. The request is sent to the NHS system.
Alternative Flows:	1. ConnectionError

Name:	RequestRepeatPrescription: ConnectionError
ID:	UC19.1
Brief Description	The NHS System did not receive repeat prescription request that has been submitted.
Primary Actor(s):	Patients.
Secondary Actor(s):	None.
Pre-conditions:	1. The Patient has submitted a repeat prescription request.
Alternative Flow:	1. The alternative flow starts after step 3 of the main flow. 2. The MHMP System fails to communicate with the NHS System due to network issues, system downtime, etc. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	SynchoniseData
ID:	UC20
Brief Description	The MHMP System synchronises patient data such as GP notes and prescription data from the NHS System.
Primary Actor(s):	Users (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals, Patients).
Secondary Actor(s):	NHS System.
Pre-conditions:	1. The MHMP System is properly authenticated and authorised to access data from the NHS System. 2. The User has been authorised to access the relevant information.
Main Flow:	1. The use case starts when a User views, creates or updates a Patient's GP notes or prescription data. 2. If they wish to view the data, then 2.1 The MHMP System sends a data request to the NHS System. 2.2 The NHS System process the request and send a response back to the MHMP System. 2.3 The MHMP System process the response data and update database if necessary. 2.4 The MHMP System display updated information to the user.
Post-conditions:	1. Data between the MHMP System and the NHS System is synchronised.
Alternative Flows:	1. RequestDenied 2. ConnectionFailure

Name:	SynchoniseData: RequestDenied
ID:	UC20.1
Brief Description	The data retrieval has been denied by the NHS System.
Primary Actor(s):	Users (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals, Patients).
Secondary Actor(s):	NHS System.
Pre-conditions:	1. The MHMP System has sent a data request to the NHS System.

Alternative Flow:	1. The alternative flow starts after step 2.1 of the main flow. 2. The NHS System detects an issue with authentication, authorisation, or patient identifiers. 3. The NHS System sends a response back to the MHMP System indicating that the data retrieval request has been denied. 4. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	SynchoniseData: ConnectionFailure
ID:	UC20.2
Brief Description	The MHMP System failed to send data to the NHS database.
Primary Actor(s):	Users (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals, Patients).
Secondary Actor(s):	NHS System.
Pre-conditions:	1. The MHMP System has attempted to send the updated data to the NHS System.
Alternative Flow:	1. The alternative flow starts after step 2.1 of the main flow. 2. The MHMP System fails to send the new or updated data to the NHS System due to connection issues. 3. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	SendMessages
ID:	UC21
Brief Description	System Administrators and Healthcare Professionals send and receive messages with one another through the MHMP.
Primary Actor(s):	System Administrators, Healthcare Professionals (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals).
Secondary Actor(s):	System Administrators, Healthcare Professionals (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals).
Pre-conditions:	1. The System Administrator or Healthcare Professional must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the System Administrator or Healthcare Professional navigates to the 'Your Messages' page on the MHMP. 2. If they choose to start a new chat, then 2.1 They select the option to compose a new message. 2.2 They search and select for a target recipient. 3. If they choose to continue an ongoing conversation, then 3.1 They search and select a conversation from the list of ongoing conversations. 4. They compose and send the message. 5. The MHMP System creates the message record in the MHMP database. 6. The MHMP System notifies the message to the recipient. 7. They exit the 'Your Messages' page or log out.
Post-conditions:	1. The message record is created in the MHMP database. 2. A User is notified of the message received.
Alternative Flows:	1. DeliveryFailure

Name:	SendMessage: DeliveryFailure
ID:	UC21.1
Brief Description	The message failed to be delivered due to a system error.

Primary Actor(s):	System Administrators, Healthcare Professionals (Psychiatrists, Therapists, Nurses, Hospital Pharmacists, Secretarial Professionals).
Secondary Actor(s):	None.
Pre-conditions:	1. The System Administrator or Healthcare Professional has composed and sent a message.
Alternative Flow:	1. The alternative flow starts after step 4 of the main flow. 2. The MHMP System fails to send the message to the recipient. 3. The MHMP System notifies them of the failure and provides an option to retry sending the message. 4. The MHMP System generates an error message.
Post-conditions:	1. An error message is generated.

Name:	ReviewSystemLogs
ID:	UC22
Brief Description	System Administrators can view system logs within the MHMP System.
Primary Actor(s):	System Administrators.
Secondary Actor(s):	None.
Pre-conditions:	1. The System Administrator must be logged onto the MHMP System.
Main Flow:	1. The use case starts when the System Administrator navigates to the 'Review System Logs' page within the MHMP. 2. The system displays a list of system logs. 3. They can filter or search the logs by date, type, severity, or other relevant criteria to isolate specific incidents. 4. They exit the 'Review System Logs' page or log out.
Post-conditions:	None.
Alternative Flows:	None.

Chapter 4

Object-Oriented Analysis and Design models

4.1 Overview

The unified modelling language, UML, is utilised to enable the transition from conceptual understanding of the MHMP, to detailed system design. This process starts with an Analysis Class Diagram that defines initial classes based on requirements and use cases. The Design Class Diagram then adds specificity, outlining system architecture and how components will function. Sequence Diagrams illustrate object interactions for key scenarios and are consistent with the Design Class Diagram, while State Machine Diagrams trace object state changes. Activity Diagrams chart the procedural flow for business cases, and Component Diagrams detail the system's modular component structure and interfacing. Finally, the Deployment Diagram emphasises how the system will be implemented at the hardware level. This progressive, diagrammatic approach streamlines the progression from abstract requirements to concrete implementation details, ensuring the design aligns with stakeholder needs and forming a robust architectural foundation for software development [15] [16].

4.2 Entity-control-boundary Pattern

During the Object-Oriented Analysis phase, the ECB (Entity-Control-Boundary) pattern was employed as a structured approach to define initial classes, providing a foundation for the transition into the Object-Oriented Design phase. Utilising the ECB pattern facilitated the refinement of diagrams into an implementation-ready solution, effectively representing and constructing a complex Object-Oriented system. The ECB pattern distinguishes three specific class types, the understanding of which is crucial for comprehending the class diagrams in their entirety.

1. **Entity Classes:**
    - Entity classes, denoted with the `<<entity>>` stereotype, define the structure and attributes of the real-world entities they represent. They serve as well-defined data containers that are constructed from retrieved records, enabling smooth data manipulation and information parsing.

2. **Control Classes:**
    - Control classes, denoted with the `<<control>>` stereotype, encapsulate the business logic and operations of the system. They serve as the backbone of the application's functionality, orchestrating the flow of data and interactions within the system.
    - These classes coordinate the interaction between multiple entity classes and manage the communication between entity and boundary classes. They play a crucial role in maintaining the consistency and integrity of the system's state. This involves updating data records in the database and ensuring the corresponding object entity is updated or instantiated as needed.
3. **Boundary Classes:**
    - Boundary classes, denoted with the `<<boundary>>` stereotype, act as interfaces between the system and external entities, such as users or other systems.

4.3 Analysis Class Diagram

The OOA Class diagram serves as a high-level overview of the system's architecture, representing the structure of the system's problem domain. The analysis was carried out utilising the Entity-Control-Boundary (ECB) pattern to structure classes.

The entity classes encapsulate the system's key data points and behaviors as identified from the domain model and use case analyses. These entities, such as *User*, *Patient*, and *HealthcareProfessional*, are designed with inheritance to capture the hierarchical relationships and shared characteristics that were identified early in the analysis phase of the project.

A central control class, *Service*, acts as a mediator that orchestrates interactions between the entities and the boundary classes. This design choice not only simplifies the system's architecture by centralising shared business logic but also enhances the system's adaptability for future expansions or modifications.

Boundary classes in the diagram, such as *UI*, *DatabaseService*, and *NHSService*, are explicitly defined to manage the system's interactions with external actors and services. These classes are crucial for the system's data persistence and user interface management, ensuring that the system can communicate effectively with users and external data sources.

The diagram illustrates the relationships between classes, using associations such as aggregations, cardinality, and inheritance to show the complex interactions within the system. The OOA Class diagram focuses on the general structure and relationships rather than detailing the specifics of implementation. This approach lays a solid foundation for transitioning to the Design Class phase, where these abstractions are refined further. *Note: attributes for entityID, creation/update details are displayed separately for readability, for the full diagram, please refer to Appendix,* ***Figure 8.2.3.1***. The dotted lines represent the `<<uses>>` dependant relationship.

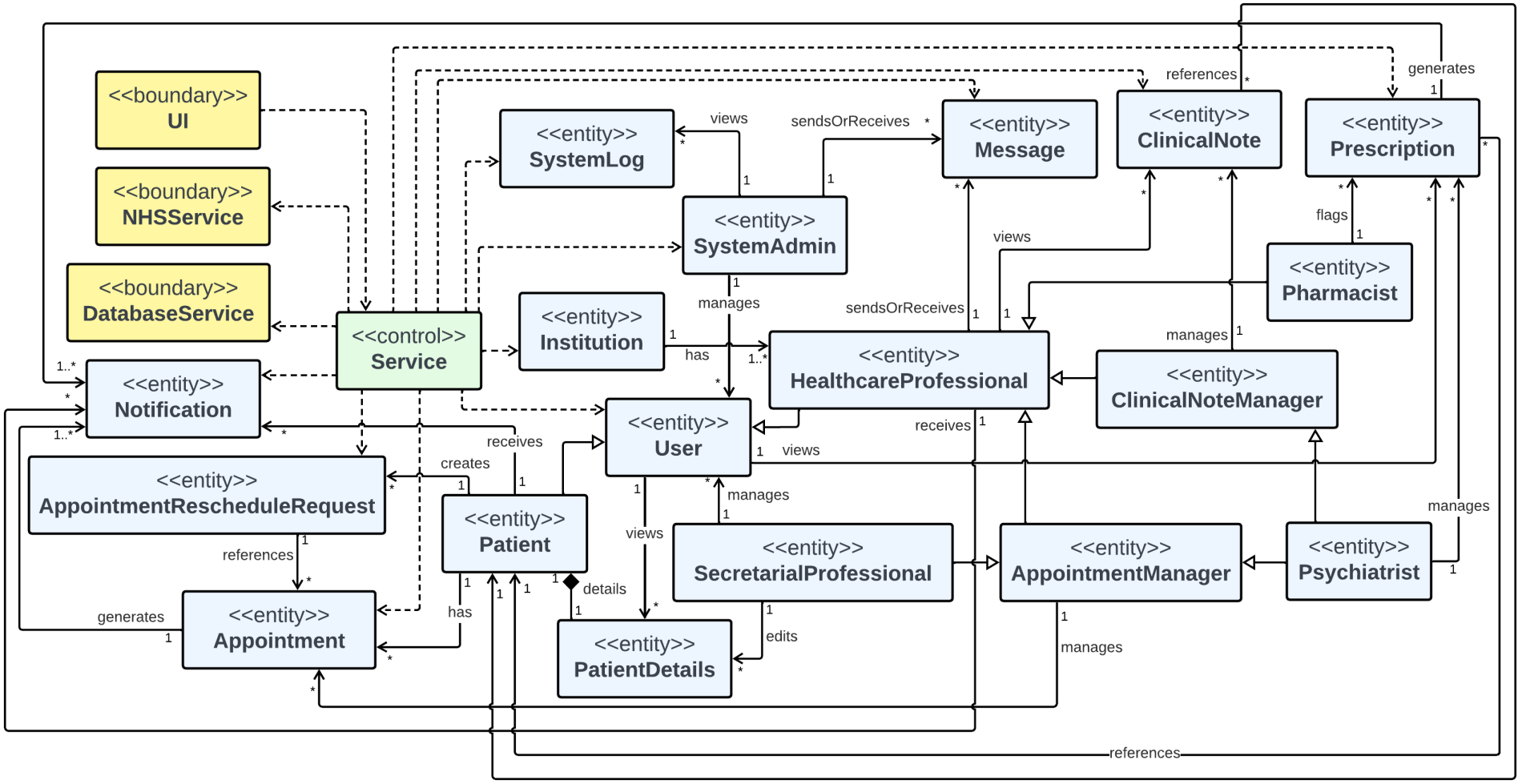


Figure 4.3.0.1: Analysis Class Reduced Diagram. Included to facilitate understanding of *Figure 4.3.0.2*

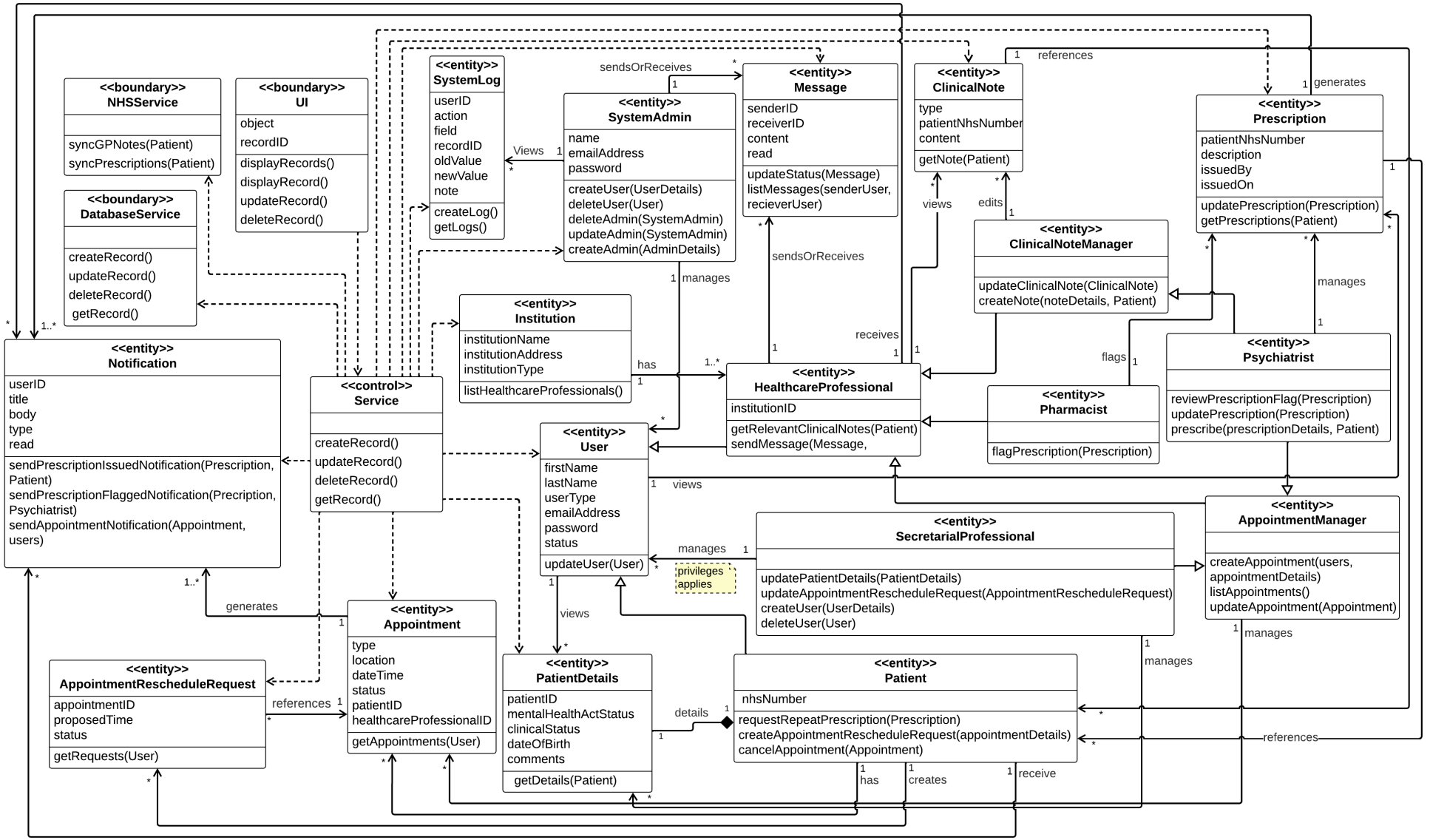


Figure 4.3.0.2: Analysis Class Diagram

4.4 Design Class Diagram

Building upon the OOA class diagram and the ECB pattern, the object-oriented design transforms the general concepts identified in the OOA phase into detailed, implementable designs. This transition involves identifying opportunities for further abstraction and consolidating conceptual methods into service classes adhering to the Single Responsibility Principle (SRP) [17]. Such classes encapsulate implementation details, facilitating a refined mapping from the problem domain into the solution domain.

4.4.1 Transition Breakdown

1. **Control Classes:**
   Control classes emerged as an important element in Object-Oriented Design (OOD), orchestrating interactions across multiple entity classes whilst managing communication between entity and boundary classes. They play an essential role in upholding the system's state's consistency and integrity, which includes updating data records in the database and ensuring the corresponding object entity is updated or instantiated as necessary. Guiding the structure and refinement of the system, control classes led to the extraction of non-basic class methods into classes adhering to the Single Responsibility Principle (SRP). This process evaluated the necessity of the resulting entity classes based on their possession of distinct attributes. Where entities did not meet this criterion, they were amalgamated into a superclass, differentiated by a type attribute to maintain their distinctiveness.

   For instance, within the context of Object-Oriented Analysis (OOA), methods from classes such as ClinicalNoteManager, AppointmentManager, and Psychiatrist were aggregated into corresponding service classes - namely, ClinicalNoteService, AppointmentService, and PrescriptionService. This consolidation led to the observation that the resulting classes lacked distinct attributes, prompting their integration into the HealthcareProfessional superclass.

2. **Entity Classes:**
   Following the structured approach adopted in Object-Oriented Design (OOD), entity classes have been refined to solely encompass data, devoid of any operational logic. This evolution aligns with the consolidation strategies applied in the control classes phase, where methods were centralised into service classes to uphold the Single Responsibility Principle (SRP). Moreover, as part of the streamlining process, the PatientDetails class was integrated into the Patient class, exemplifying the methodology's efficacy in reducing the overall number of entity classes. This refinement ensures that entity classes now function as pure data holders, contributing to a more coherent and simplified system structure that mirrors the disciplined approach taken with control classes.

3. **Boundary Classes:**
   During the initial analysis phase, the necessity for the NHSService was identified in order to interact with an external data system. The DatabaseService was identified as necessary for data persistence and the UI was required to provide an interface with the end user. These boundary classes are relatively immutable within our system, as they are required regardless of the implementation details. Nevertheless, the methods contained in the boundary classes are detailed in a higher level view, allowing greater flexibility in the implementation / integration phase of development. The differences in the boundary classes between the OOA and the OOD are simply the parameters of the methods contained within the classes, and the number of associations present due to the increase in the number of service classes.

4.4.2 System Flow Overview

The system operates through a streamlined flow where the UI boundary class initiates user requests, which are processed by control classes using business logic. These control classes interact with entity classes for data operations and with other boundary classes for additional data handling, linking the system's internal operations with external entities.

This layered architecture promotes separation of concerns, modularity, and flexibility in the system design. It allows for easier maintenance, testing, and evolution of the system components independently.

Note: For the class diagrams, we have followed UCL material (COMP1008 object-oriented programming) for our constructor representation choice [18].

Note: For notational brevity, dotted lines are used to represent the `<<uses>>` **dependent relationship. [14]**

Note: attributes for entityID, creation / update details and the methods of constructor, getters and setters are displayed separately for readability in *Figure 4.4.2.3*.

Note: For list values (enumerations) for the attributes presented in the OOD, (status, type and clinical status), please refer to *Appendix - Subsection 8.2.2*

For the full diagram PlantUML code, please refer to ***Appendix - Subsection 8.2.3***.

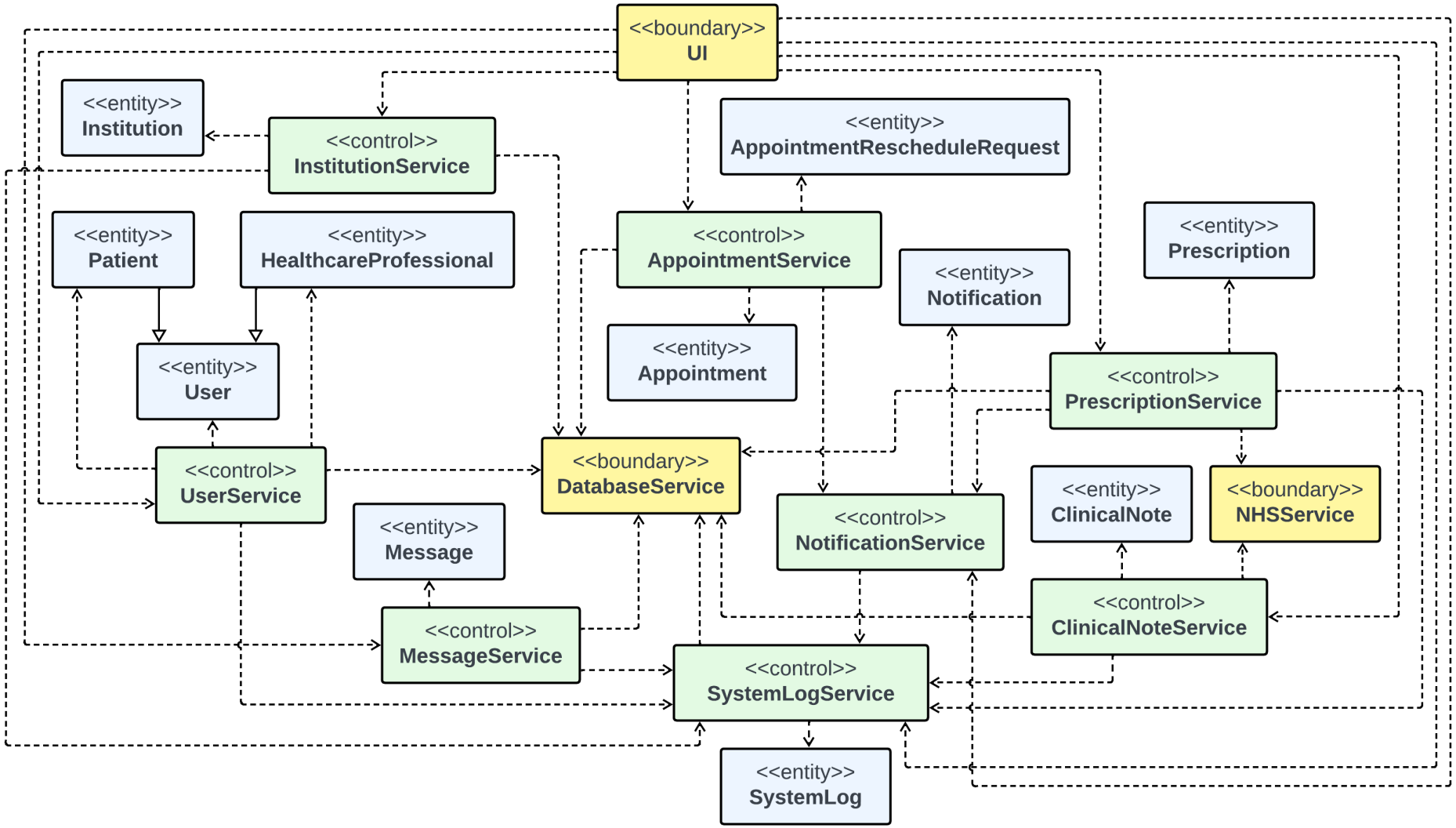


Figure 4.4.2.1: Design Class Reduced Diagram. Included to facilitate understanding of *Figure 4.4.2.2*

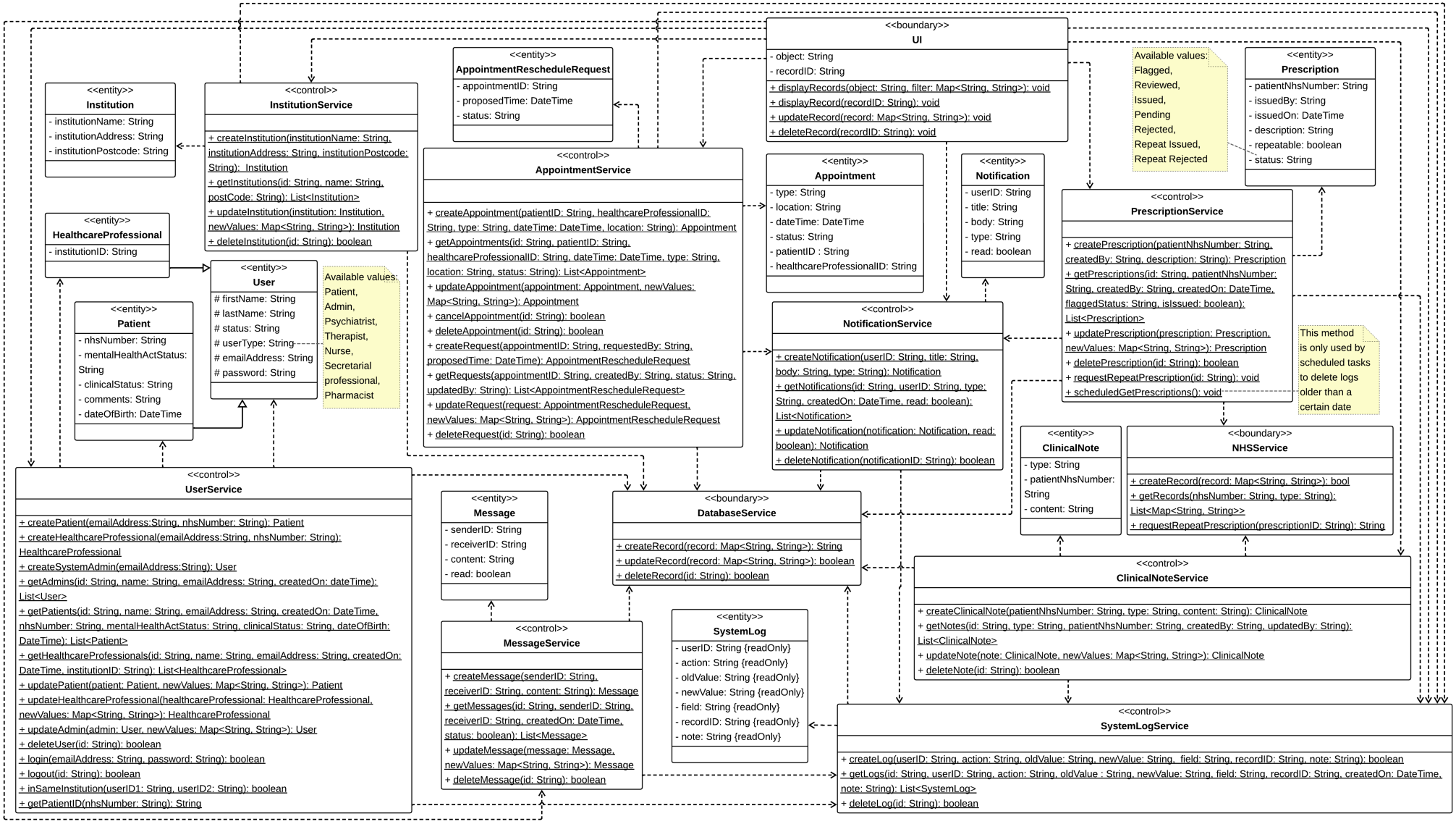


Figure 4.4.2.2: Design Class Diagram

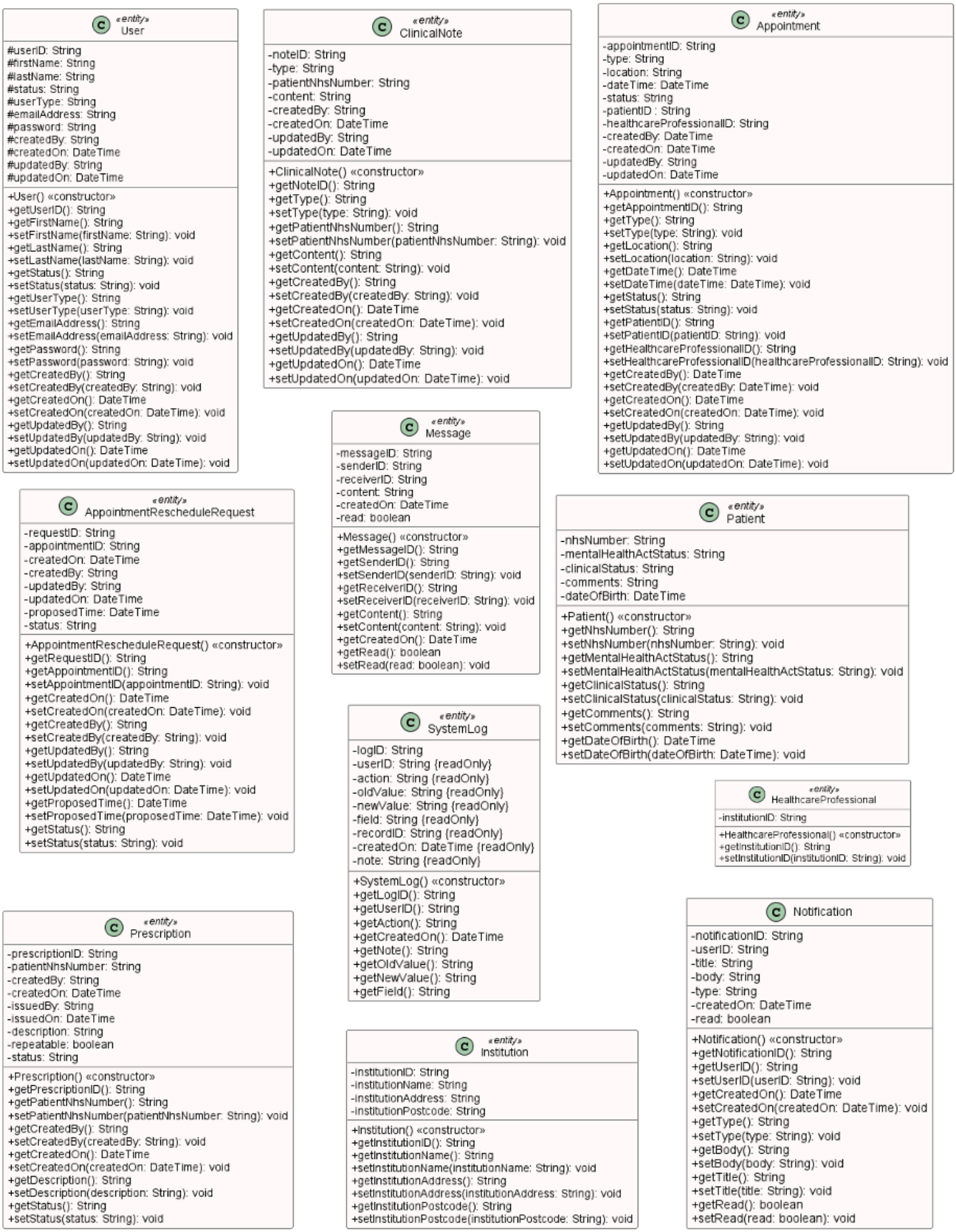


Figure 4.4.2.3: Full Entity Class Listings with All Methods and Attributes Listed

4.5 Sequence Diagrams

Sequence diagrams show the interactions between objects or components in a software system over a timeline (shown by the lifeline for actors and participants), and showing the flow of messages exchanged among them in order to execute a sequence of events within a task. Our design justifications are as follows:

- Use of <<create>> and <<destroy>> [19]
- Use of 'ref' [20]
- Use of variable assignment [21]
- Use of arrow notation [22]

4.5.1 Sequence 1: Create System Log

Figure 4.5.1.1 depicts the sequence for the creating of a system log. The purpose of the system log is to store a record of all of the MHMP's activity. Since this is an automated process, we do not start with an external actor. The *createLog()* method is called from the **SystemLogService** class taking various inputs to record a log. The *createRecord()* method from the **DatabaseService** class is then called, which stores the data in the MHMP database. This sequence is carried out after any database execution. However, for readability, the 'ref' call to this sequence is omitted from other sequence diagrams.

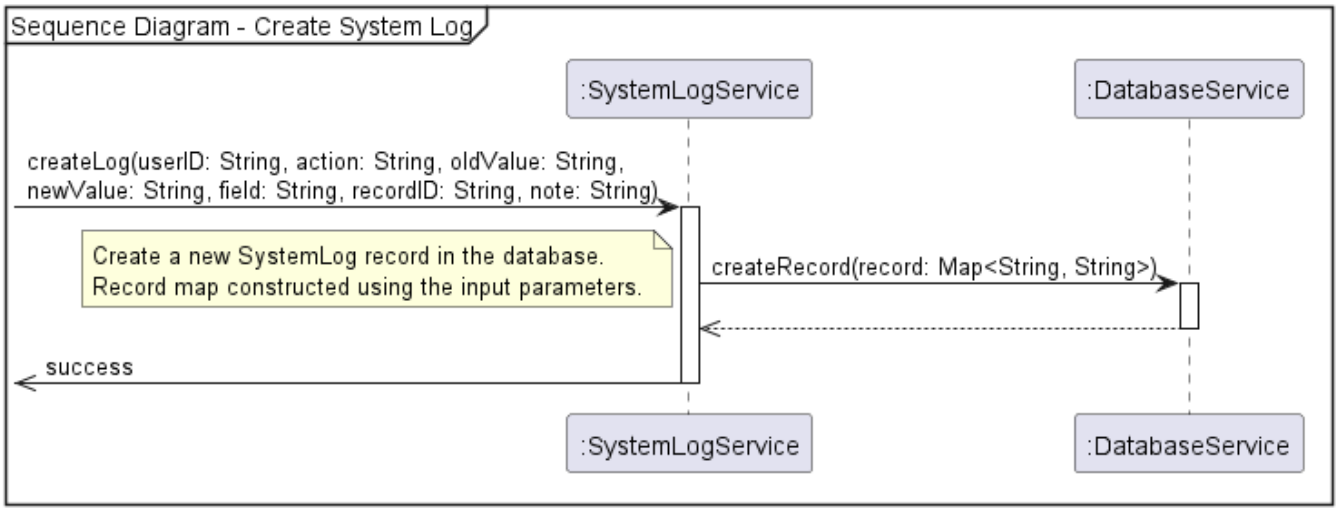


Figure 4.5.1.1: Automated system log creation

4.5.2 Sequence 2: Send Notification

Figure 4.5.2.1 depicts the sequence for the MHMP sending a notification to the system user. The sequence starts with the method *createNotification()* from the **NotificationService** class, taking inputs for the notification content. The *createRecord()* method from the **DatabaseService** class is then utilised to store the notification content. An instance of the **Notification** class is then created (*n1*), which is then displayed in the UI and viewed by the user.

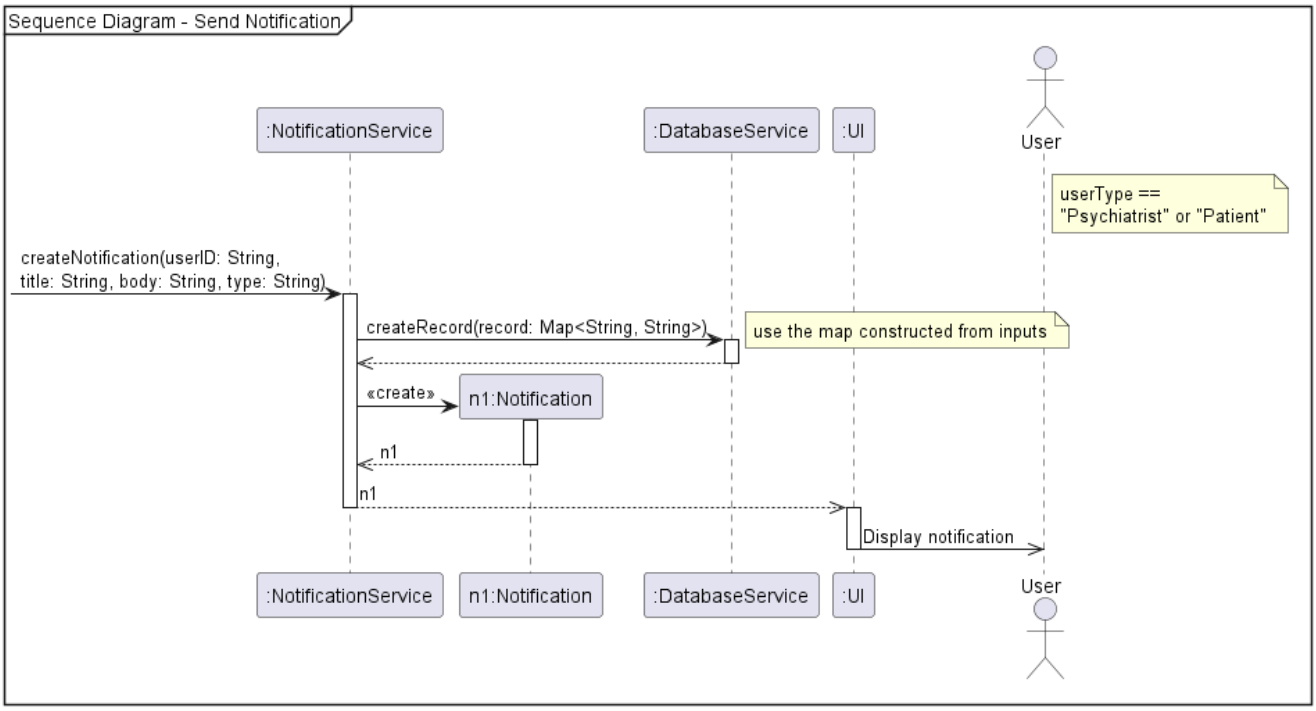


Figure 4.5.2.1: MHMP sending notification

4.5.3 Sequence 3: Request Appointment Reschedule

Figure 4.5.3.1 depicts the sequence for the patient requesting a rescheduled appointment. When the patient submits this request form, the method *createRequest()* from the **AppointmentService** class is utilised. This method takes the parameters 'appointmentID' (from an aready existing appointment record), 'requestedBy', and 'proposedTime'. The *createRecord()* method from the **DatabaseService** class is then used to record a map constructed from the input parameters. An instance of **AppointmentRescheduleRequest** (*r1*) is then created, and displayable in the UI, generating a success message for the patient.

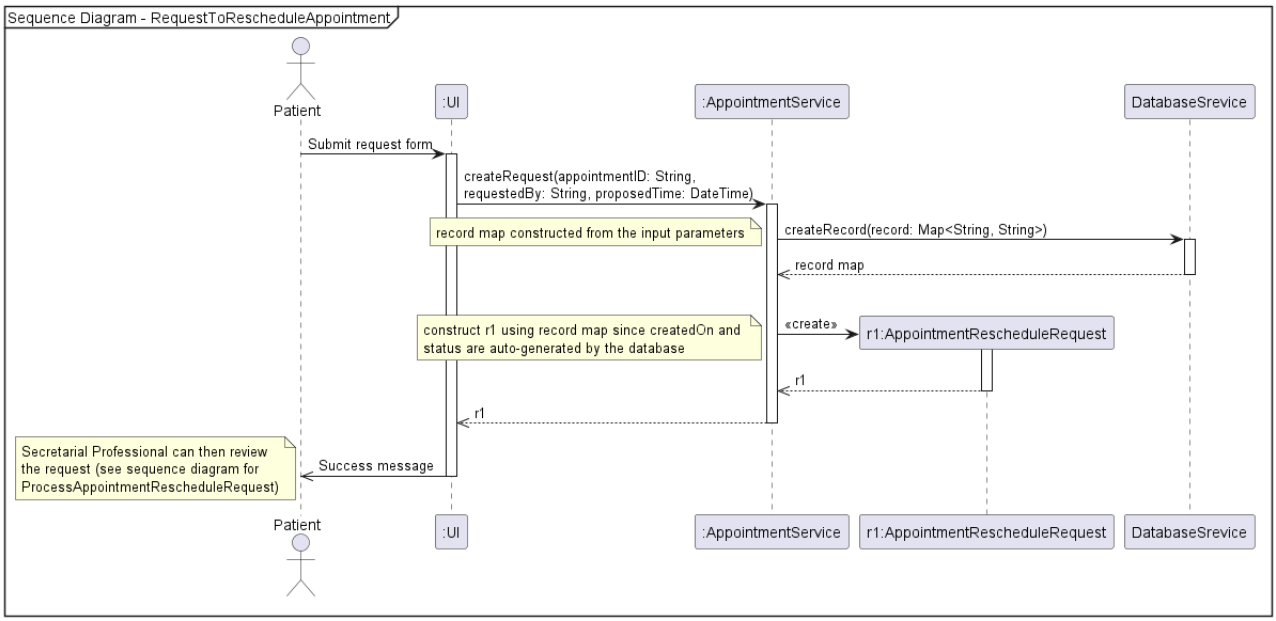


Figure 4.5.3.1: Patient request to reschedule an appointment

4.5.4 Sequence 4: Process Appointment Reschedule Request

Figure 4.5.4.1 depicts the sequence for the secretarial professional (healthcare professional) processing a reschedule request. The sequence initiates with the secretarial professional updating the request through the UI. The *updateRequest()* method from the **AppointmentService** class is used with the parameter 'request' set to *r1*, an instance of the **AppointmentRescheduleRequest** class. The *getRequestID()* method from the **AppointmentRescheduleRequest** class is then used to retrieve the request ID (*rID*). The *updateRecord()* method from the **DatabaseService** class is then utilised to update the record. A loop occurs and the corresponding *setter()* method from the **AppointmentRescheduleRequest** class is then used for each value in the record map. The *getAppointmentID()* method from the same class is then used to retrieve the appointment ID (*aID*) used in the initial request.

Now that the **AppointmentService** class has retrieved the *aID*, the *getAppointments()* method in this class is then utilised to retrieve the appointment information. This is achieved by a self call with the *getAppointments()* method with its 'appointmentID' parameter set to *aID*. This retrieves the instance *a1* of the **Appointment** class. In order to update the appointment time, the *updateAppointment()* method from the **AppointmentService** class is used on a self call to update the appointment 'dateTime' and change the 'updatedBy' appointment attribute. The change is recorded by the *updateRecord()* method from the **DatabaseService** class and the record is constructed using the map used in the *updateAppointment()* method.

The next step is to produce the notification. For this to be done correctly, the healthcare professional ID and the patient ID need to be retrieved from the appointment information in *a1*. The *getHealthcareProfessionalID()* and *getPatientID()* methods from the **Appointment** class are used to return these IDs (*hcID* and *pID* respectively).

The *createNotification()* method from the **NotificationService** class is used for both sets of notifications issued, and are executed in parallel within the sequence. For the first notification to the healthcare professional, this method takes its 'userID' parameter set to *hcID*, its 'title' parameter carrying the *aID* information, and its body parameter carrying the 'newTime' which was set previously. The notification is then sent to the healthcare professional (see **Figure 4.5.2.1** for the process for sending a notification). For the second notification, the same process occurs but with the *createNotification()* method's 'userID' parameter set to *pID*. Once these notifications are sent out, the *a1* appointment instance is destroyed - denoted by the red cross. We no longer need this instance as it was created only for the *updateAppointment()* method.

Finally, the updated request (*r1*) is passed back to the UI to notify the secretarial professional the updated result.

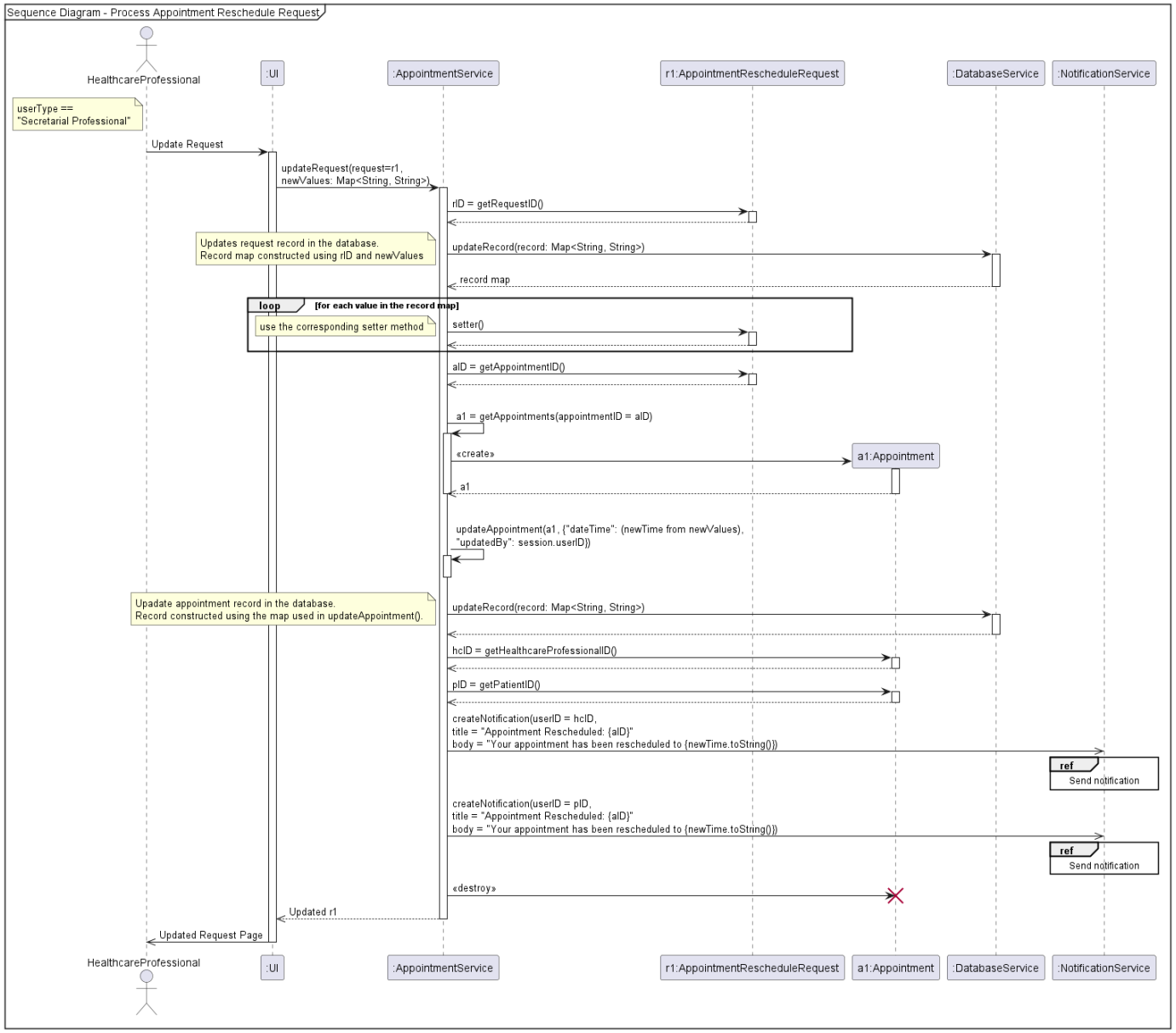


Figure 4.5.4.1: Healthcare professional processes an appointment reschedule request

4.5.5 Sequence 5: View Clinical Note

Figure 4.5.5.1 depicts the sequence for the Healthcare Professional viewing a clinical note. The process is triggered by the UI page for clinical notes, calling the method *getNotes()* from the **ClinicalNoteService** class. At this point, there are two different processes that take place depending on the note type, signalled by the 'alt' box. If the note type is of GP note, the *getRecord()* method from the **NHSService** class is utilised, taking the patient NHS number as its parameter. The **NHSService** class returns the note to the **ClinicalNoteService** class. In the other case, when the note is not of type GP note, then the *getRecords()* method from the **DatabaseService** class is instead utilised. Now, in both cases, a loop is used to create a note using the **ClinicalNote** class for each retrieved record, and so the clinical note is passed through the **ClinicalNoteService** class and displayed in the UI as a list of clinical notes.

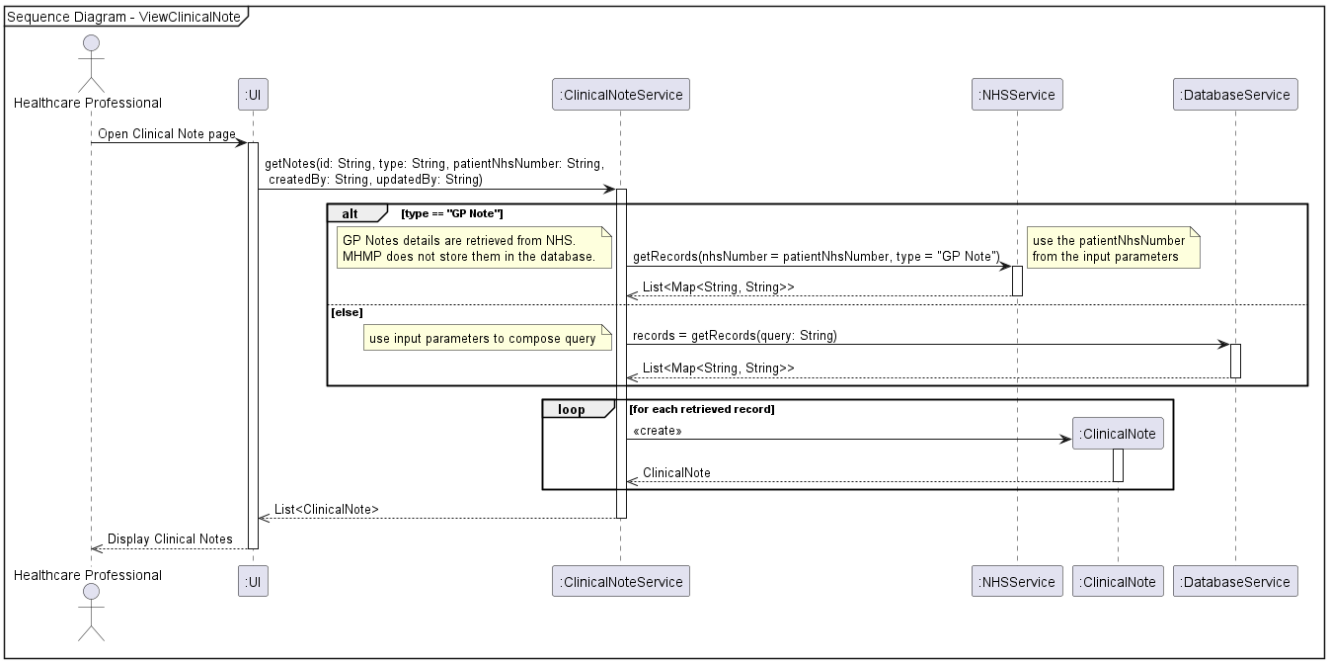


Figure 4.5.5.1: Healthcare professional views clinical note

4.5.6 Sequence 6: Update Prescription - Psychiatrist / Pharmacist

Figure 4.5.6.1 depicts the sequence for the Psychiatrist / Pharmacist updating a patient prescription. The sequence starts with the Healthcare professional of type Pharmacist or Psychiatrist needing to update the prescription. The *updatePrescription()* method from the **PrescriptionService** class is used, taking an already instantiated instance of the Prescription class (*p1*). The *getPrescriptionID()* method is utilised to retrieve the prescription ID, and the *setters()* method from the **Prescription** class is used to set each new value. The *updateRecord()* method from the **DatabaseService** class is then used to store the changed values.

At this point, there are different routes for both user types. For the Psychiatrist, once the record is updated using the *updateRecord()* method from the **DatabaseService** class, the updated prescription (*p1*) is returned from the **PrescriptionService** and back to the UI for viewing.

However, in the Pharmacist case, once the *updateRecord()* method from the **DatabaseService** class is used, they have updated the status to 'flag' for the prescription (*p1*). This means an additional sequence is executed; the 'createdByID' is obtained from the *getCreatedBy()* method in the **Prescription** (*p1*) class, and used to create a notification. The prescription notification data is stored using the *createNotification()* method from the **NotificationService** class, taking the 'createdByID' as the 'userID', and continues the "Send Notification" process as referenced in the diagram. (See **Figure 4.5.2.1** for this process.) Finally, the updated prescription (*p1*) is returned from the **PrescriptionService** and back to the UI for viewing, as in the case for the Psychiatrist.

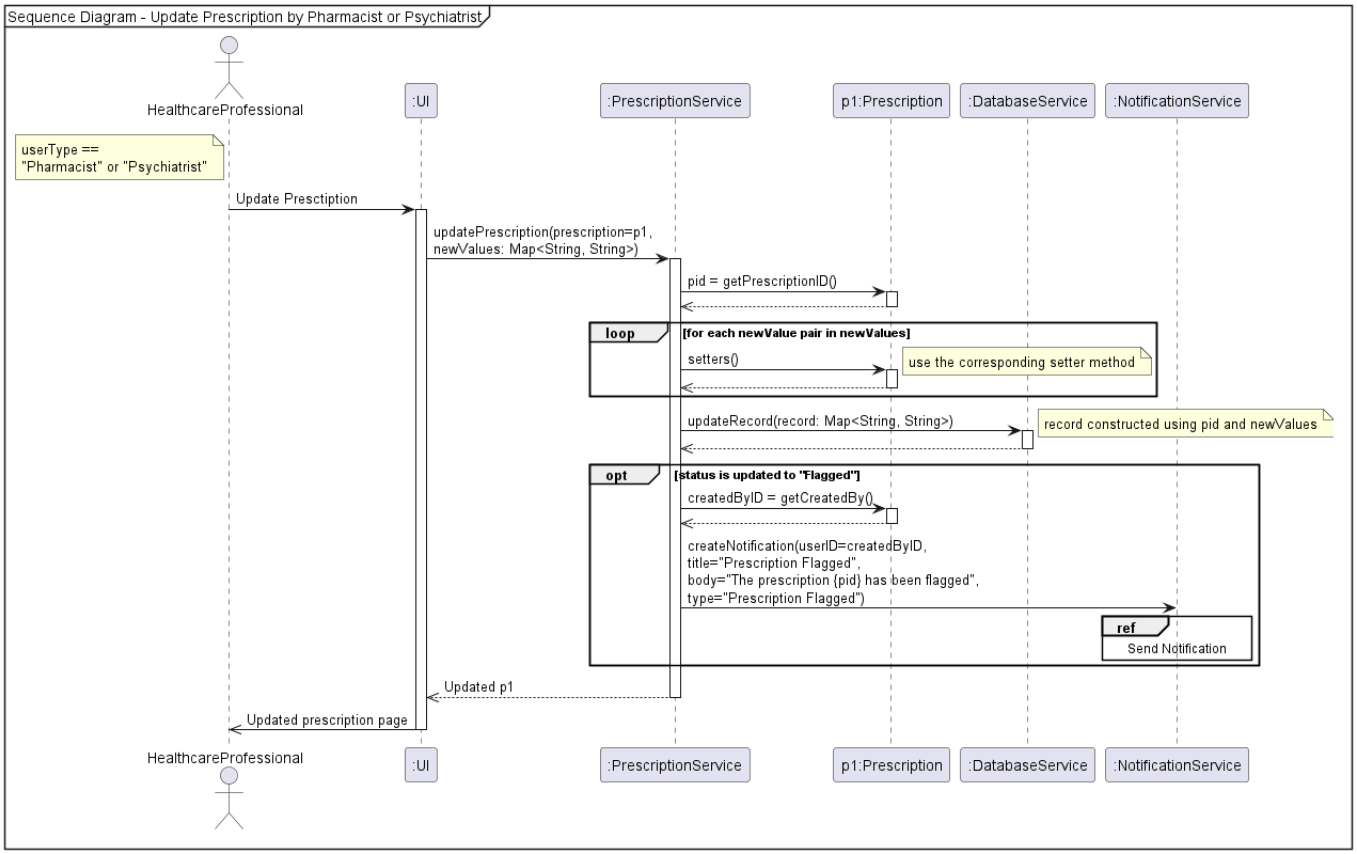


Figure 4.5.6.1: Psychiatrist or pharmacist updating a prescription

4.6 State Machine Diagrams

State machine diagrams analyse the behavior of an object within the system by defining its states, transitions between them, and the actions triggering those transitions, capturing the life-cycle. [14]

4.6.1 State Machine 1: User

Figure 4.6.1.1 depicts the life-cycle for a User. Their account starts in the 'Active' state. The User can then change state through the *deactivate* transition to an 'Inactive' state (performed by system admins or secretarial professionals). The reverse is also true, using the *activate* transition.

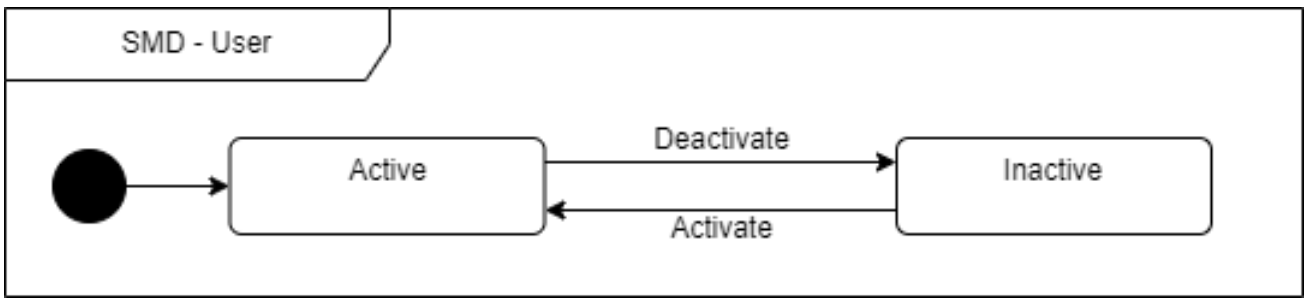


Figure 4.6.1.1: State machine for User

4.6.2 State Machine 2: Appointment

Figure 4.6.2.1 depicts the life-cycle for an Appointment. Upon creation of the appointment, its state is 'Scheduled'. Updating status from 'Scheduled' to 'Complete' is completed by a stored procedure in the database that compares the current date-time with the appointment scheduled date-time. In another scenario, the appointment is rescheduled, and remains in the 'Scheduled' state through a self transition *reschedule*. In our final scenario, the appointment can change state to 'Cancelled' through the *cancel* transition, and remains in this state.

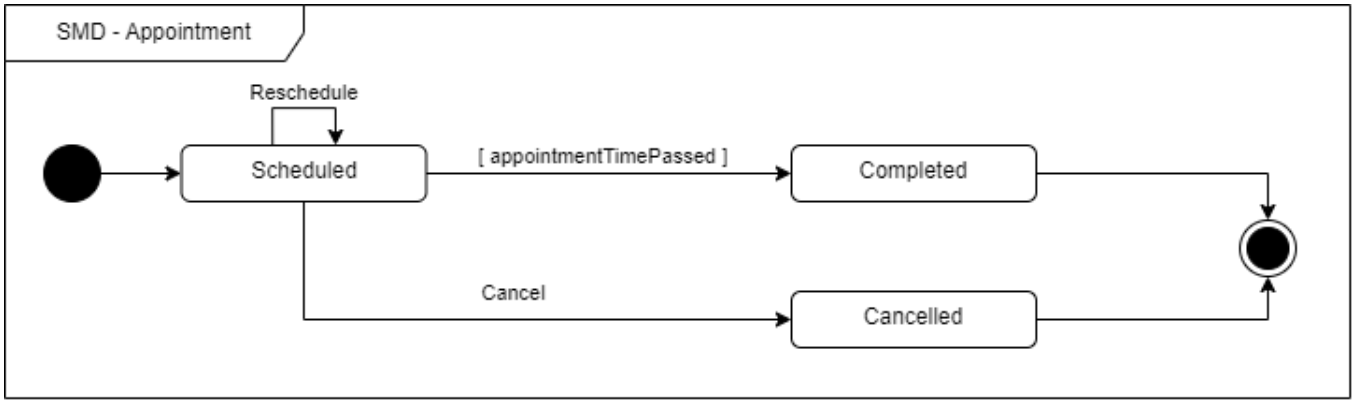


Figure 4.6.2.1: State machine for Appointment

4.6.3 State Machine 3: Appointment Reschedule Request

Figure 4.6.3.1 depicts the life-cycle for an Appointment Reschedule Request. The request starts off in the 'Pending' state and changes to the 'Resolved' state through the *resolve* transition (performed by secretarial professionals), and remains in this state.

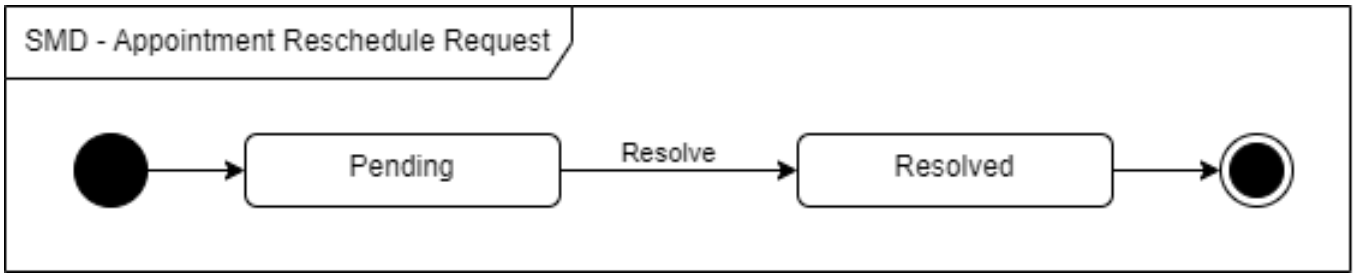


Figure 4.6.3.1: State machine for Appointment Reschedule Request

4.6.4 State Machine 4: Prescription

Figure 4.6.4.1 depicts the life-cycle for a Prescription. Since we are interacting with the NHS system, an explanation is required for each case. The simplest case is if the prescription has the state 'Rejected' and this occurs if the repeat prescription request is rejected. The prescription starts off with the 'Issued' state otherwise.

An issued prescription could have its state changed by either pharmacists flagging the prescription or by patients requesting a repeat prescription. If flagged by a pharmacist, the prescription would either terminate with the 'Flagged' state or be reviewed by the psychiatrist and terminate with 'Reviewed' state. If the patient requests a repeat prescription, the state changes into 'Pending'. Depending on the outcome of scheduledGetPrescription() from the PrescriptionService, the state will be updated to 'Repeat Issued' or 'Repeat Rejected'. From that, the prescription can be updated to 'Pending' again if requested by the patient again or be updated to 'Issued' if the 'repeatable' flag is updated to false by the psychiatrist.

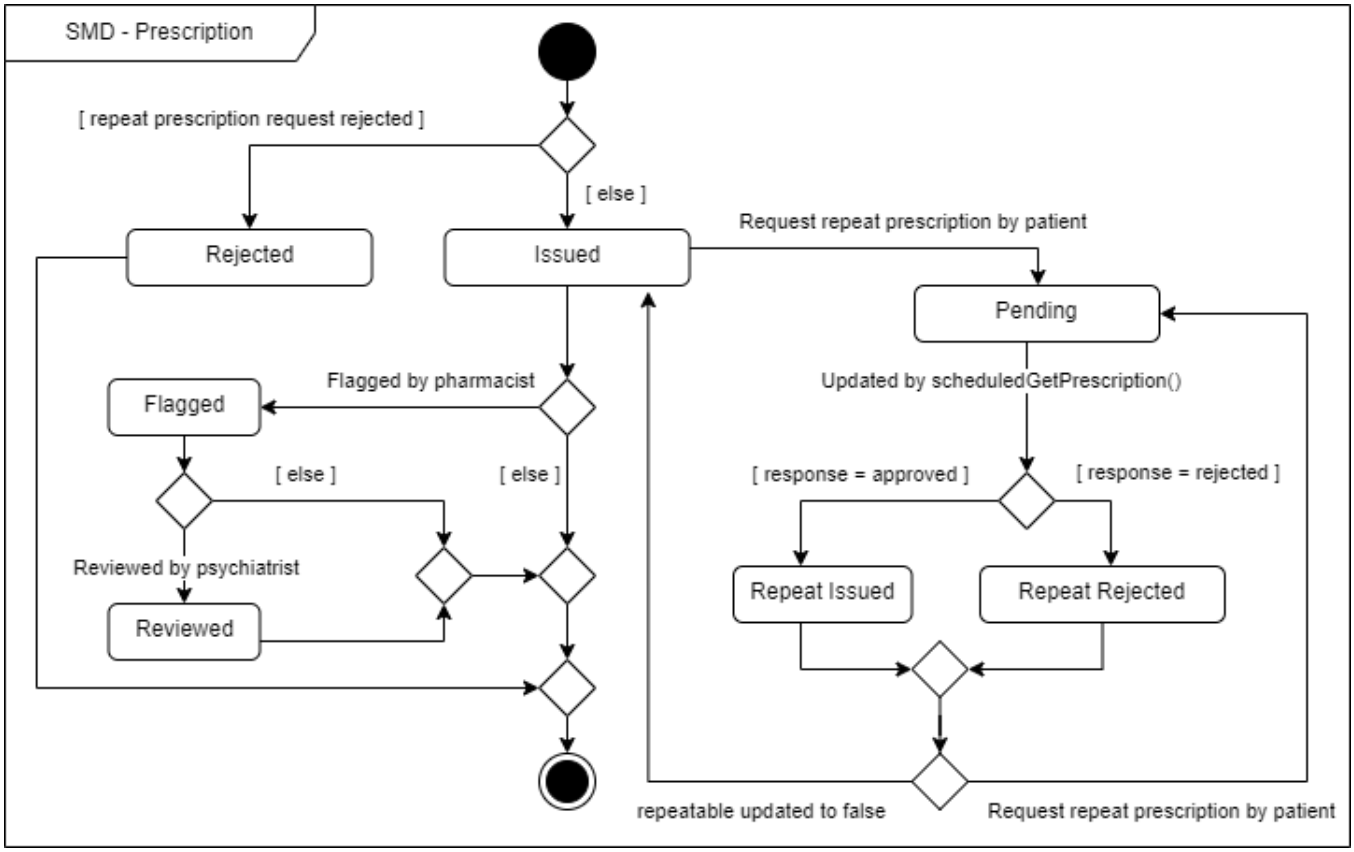


Figure 4.6.4.1: State machine for Prescription

4.7 Activity Diagrams

Activity diagrams show the flow of activities within a software system, showing the sequence of tasks and branches available for alternative paths for the activity [23]. For our system, activity cases for appointment, prescription and notes management are presented below.

4.7.1 Activity 1: Appointment Management - Appointment Manager

Figure 4.7.1.1 depicts the appointment management process initiated by a secretarial professional, therapist or psychiatrist. It starts off with the user creating or editing an appointment. The MHMP then creates / updates an appointment record, and sends a notification to the appointment participants.

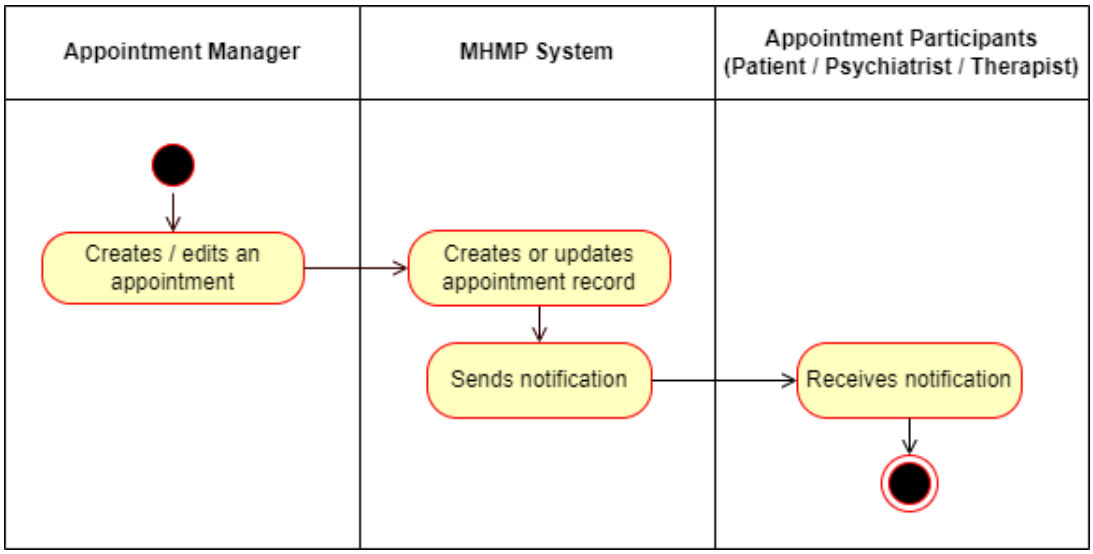


Figure 4.7.1.1: Appointment Management (Appointment creation/editing by appointment managers)

4.7.2 Activity 2: Appointment Management - Patient Request

Figure 4.7.2.1 depicts the patient reschedule request process using swimlanes for Patient, Secretarial Professional, MHMP System and Non-patient Participants. It starts off with the patient submitting a reschedule request to reschedule their appointment with a proposed time. This creates a record in the MHMP System. The Secretarial Professional then resolves the request with either the proposed time slot or their chosen next suitable time slot. This updates the request record created by the reschedule request action from the patient and updates the related appointment record. Notifications of the change will be generated for the appointment participants.

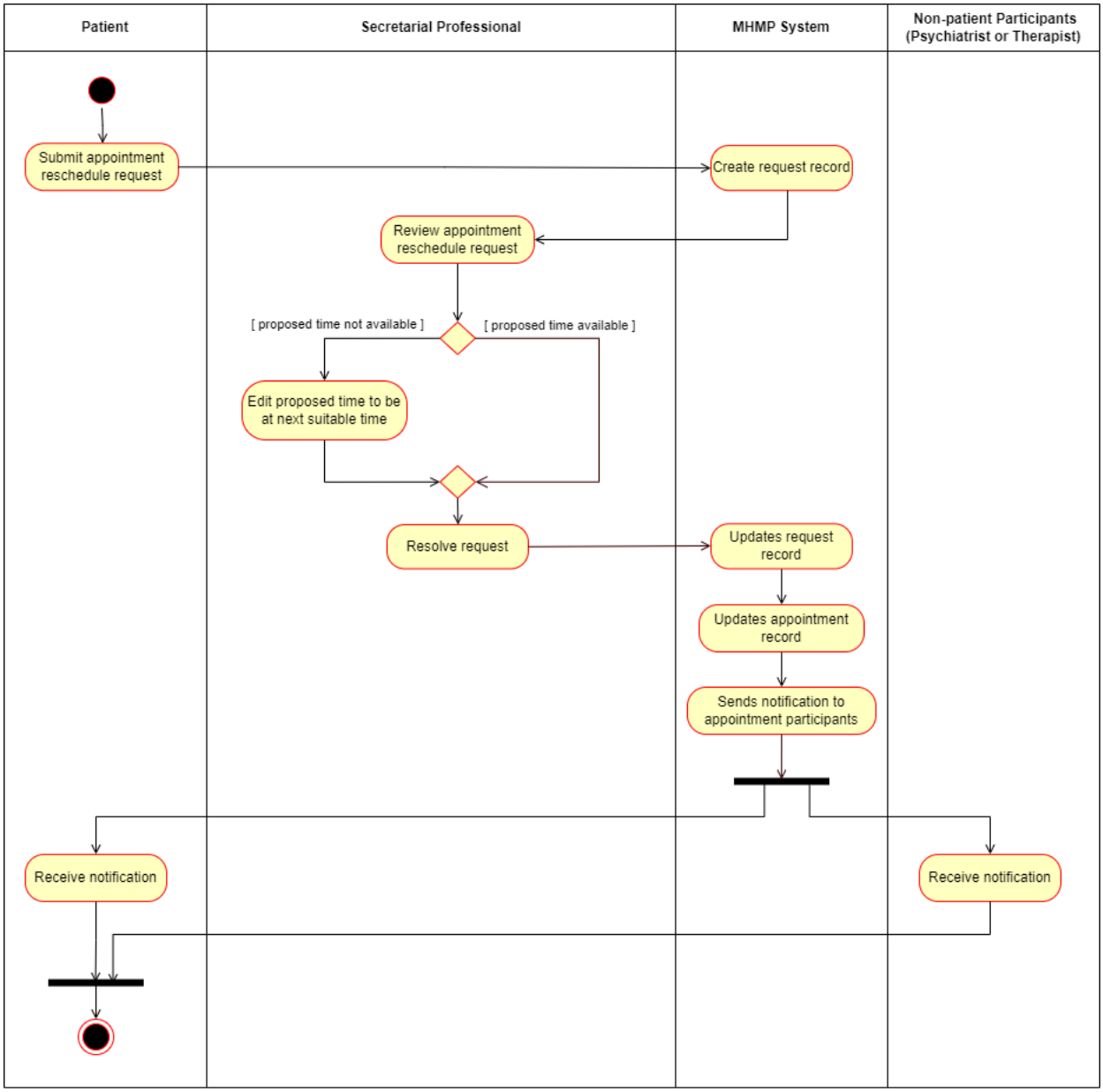


Figure 4.7.2.1: Appointment Management - Reschedule Request from Patient

4.7.3 Activity 3: View Clinical Notes

Figure 4.7.3.1 depicts the clinical note viewing process by nurses, psychiatrist and therapists. It starts off with the user opening a patient's clinical note. The MHMP uses note type to determines which process to follow. If a GP note is requested, the MHMP sends the request to the NHS system, which processes the request. A response is received by the MHMP and the system displays the clinical note information. Alternatively, if the clinical note is not of type GP, the process remains in the MHMP and queries the MHMP database. The user is then able to view the clinical note in both routes.

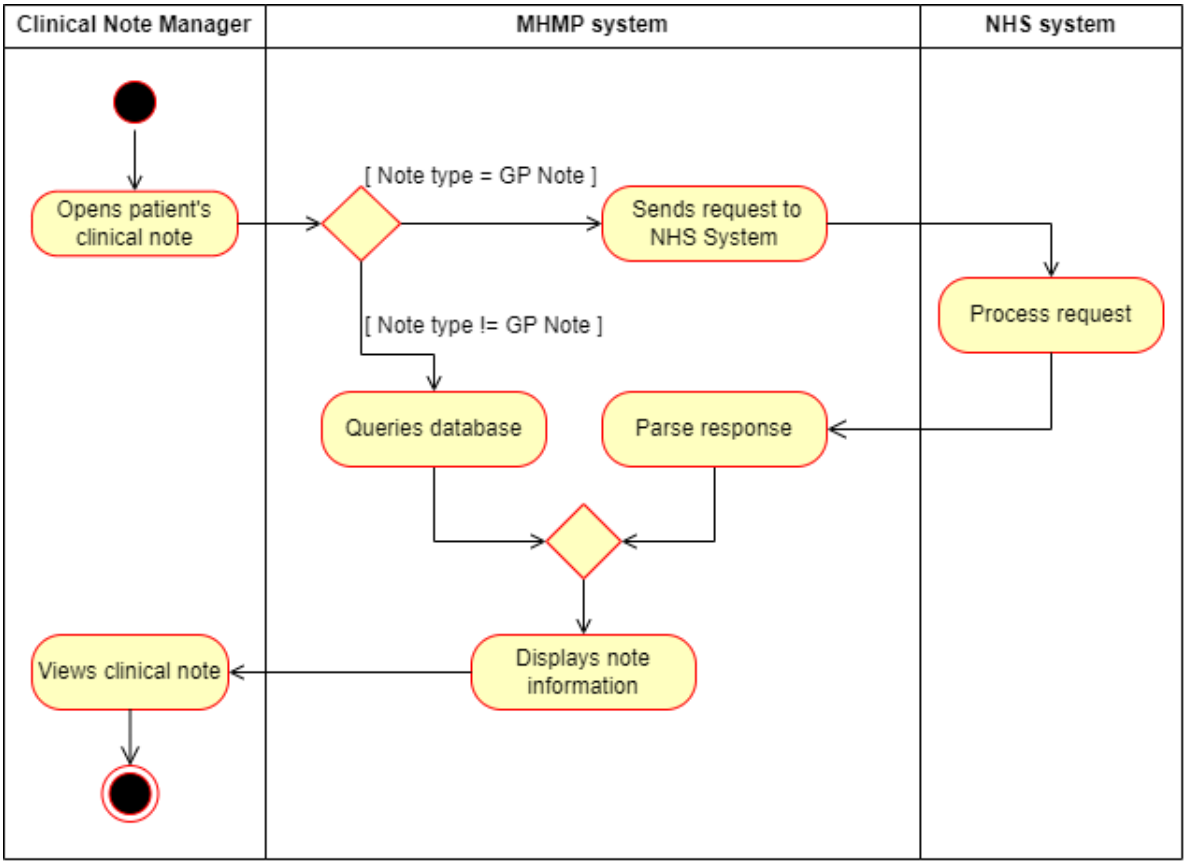


Figure 4.7.3.1: View Clinical Notes

4.7.4 Activity 4: Prescription - Psychiatrist

Figure 4.7.4.1 depicts the creation / update of a prescription by a psychiatrist process. The psychiatrist creates / updates the prescription which initiates parallel processes - the prescription record is created / updated and simultaneously the MHMP system sends a create / update request to the NHS system (if required) which parses the response once received. The parallel processes rejoin and, in the caes of creation, sends a prescription notification to the patient.

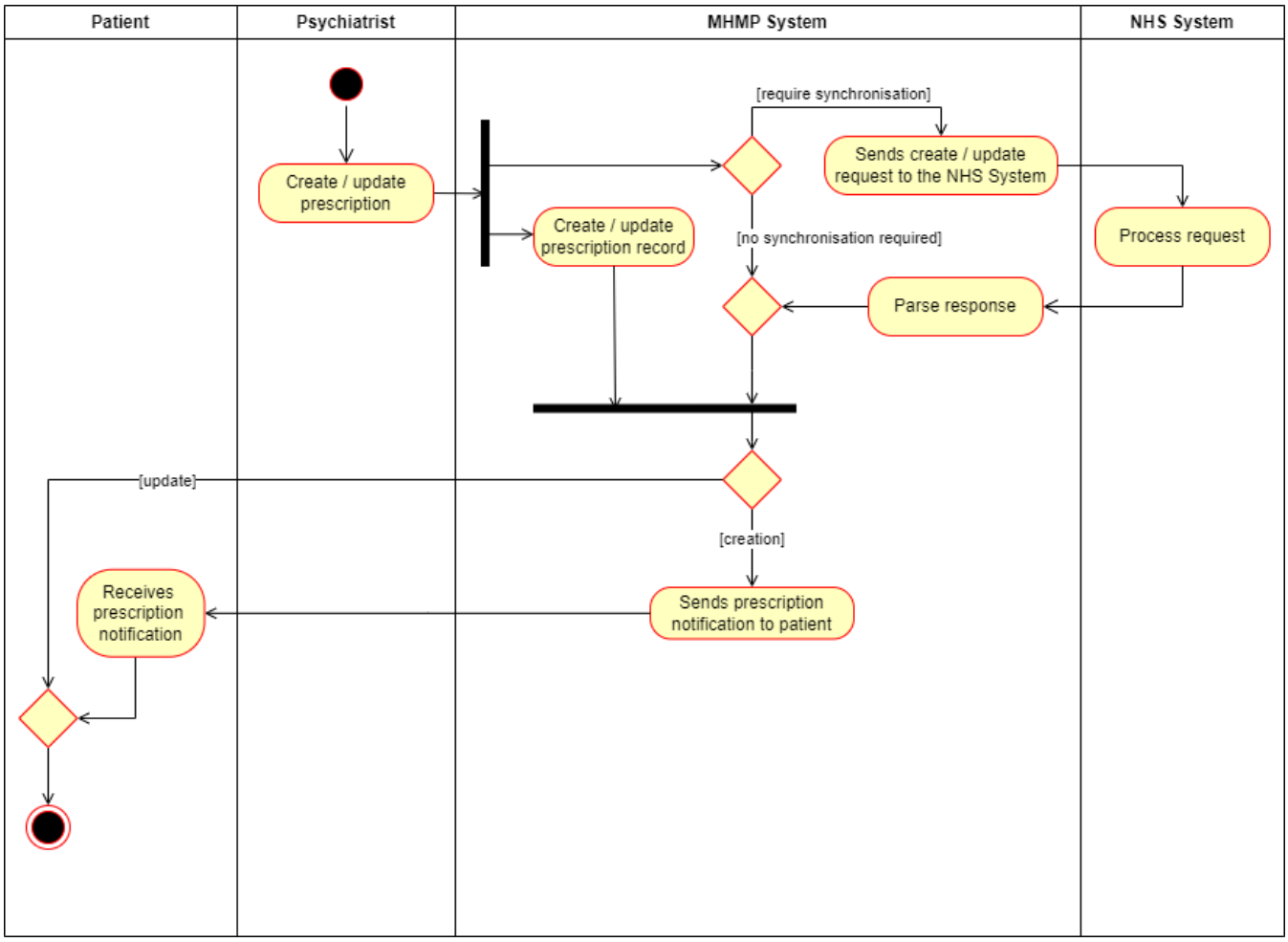


Figure 4.7.4.1: Prescription Creation or Update by Psychiatrist

4.7.5 Activity 5: Prescription - Pharmacist Flag

Figure 4.7.5.1 demonstrates a flow where a hospital pharmacist flags a prescription within the MHMP system. The pharmacist flags a prescription, which prompts the MHMP system to update the status of the prescription to 'Flagged' and then send a notification to a psychiatrist.

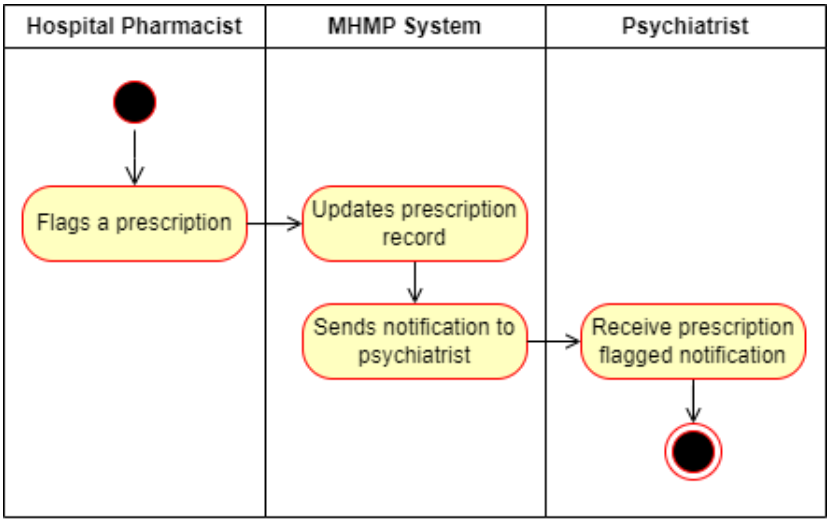


Figure 4.7.5.1: Pharmacist flags a prescription

4.7.6 Activity 6: Prescription - Patient Repeat Prescription Request

This activity diagram (**Figure 4.7.6.1**) outlines the flow for a patient requesting a repeat prescription (RP) through the MHMP. The process begins with the patient initiating a request for a repeat prescription within the MHMP. As a result, the MHMP sends the request to the NHS system and updates the prescription record in the MHMP database. These two parallel processes join and then fork again for another set of parallel processes. The MHMP updates its RP record based on the NHS System response, and a new prescription record is created based on this same response. After the these processes join, this sends out a notification to the patient.

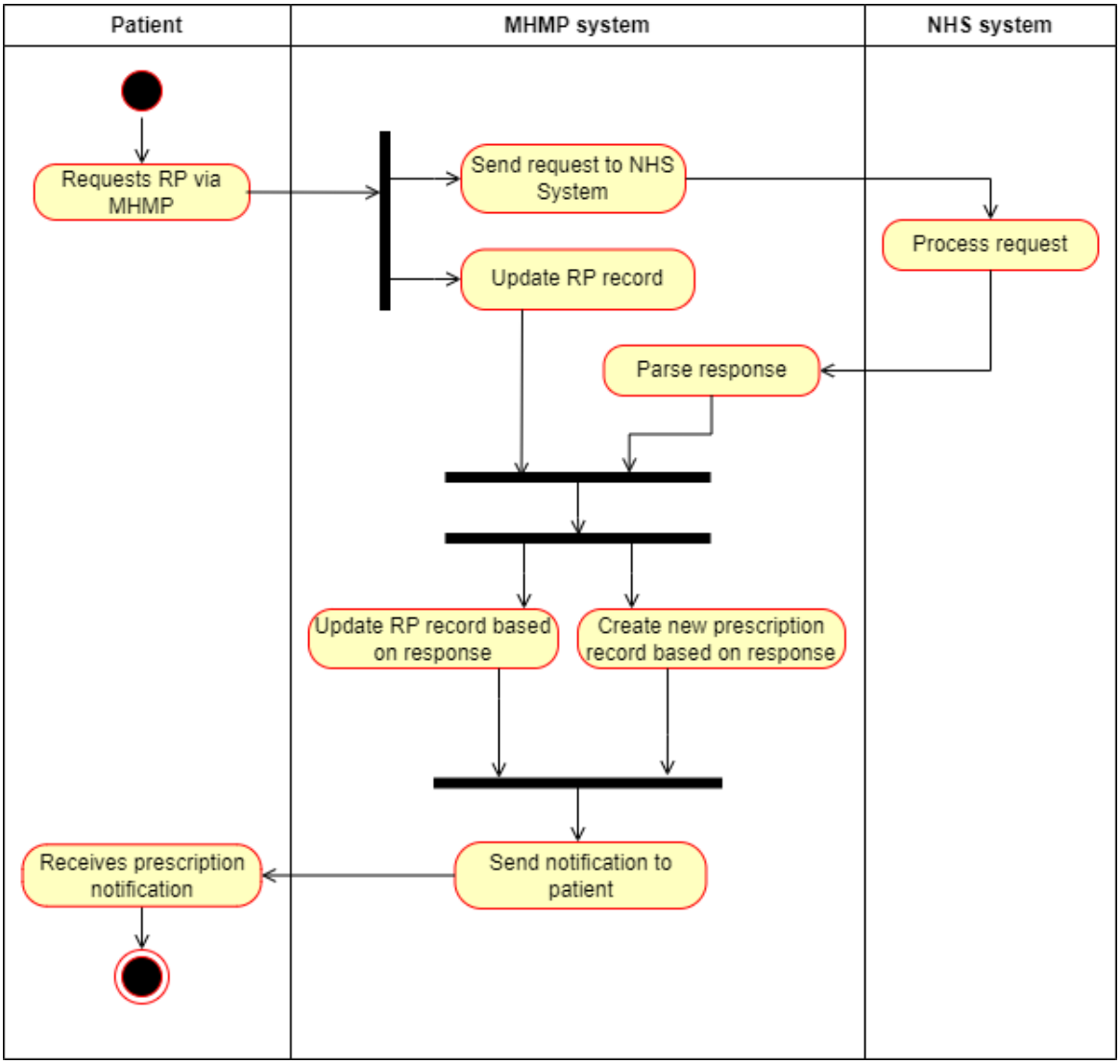


Figure 4.7.6.1: Patient requests repeat prescription (RP)

4.8 Component Diagram

Component diagrams describe the components of a software system and their relationships illustrating the organisation and dependencies of the modules in the system [14]. This component diagram (**Figure 4.8.0.1**) depicts the architecture of the MHMP. The system is modular, with a graphical user interface (UserInterface) and dedicated components for clinical data to orchestrate the flow of medical records and prescriptions (ClinicalDataManagement), interfacing with the NHS for prescription and GP notes information (NHSService), user management (UserManagement), notifications (NotificationManagement), appointments (AppointmentManagement), messaging (Messaging), system logging for the system administrator (SystemLogging), and a database component (DatabaseManagement) to handle the data persistence manipulations.

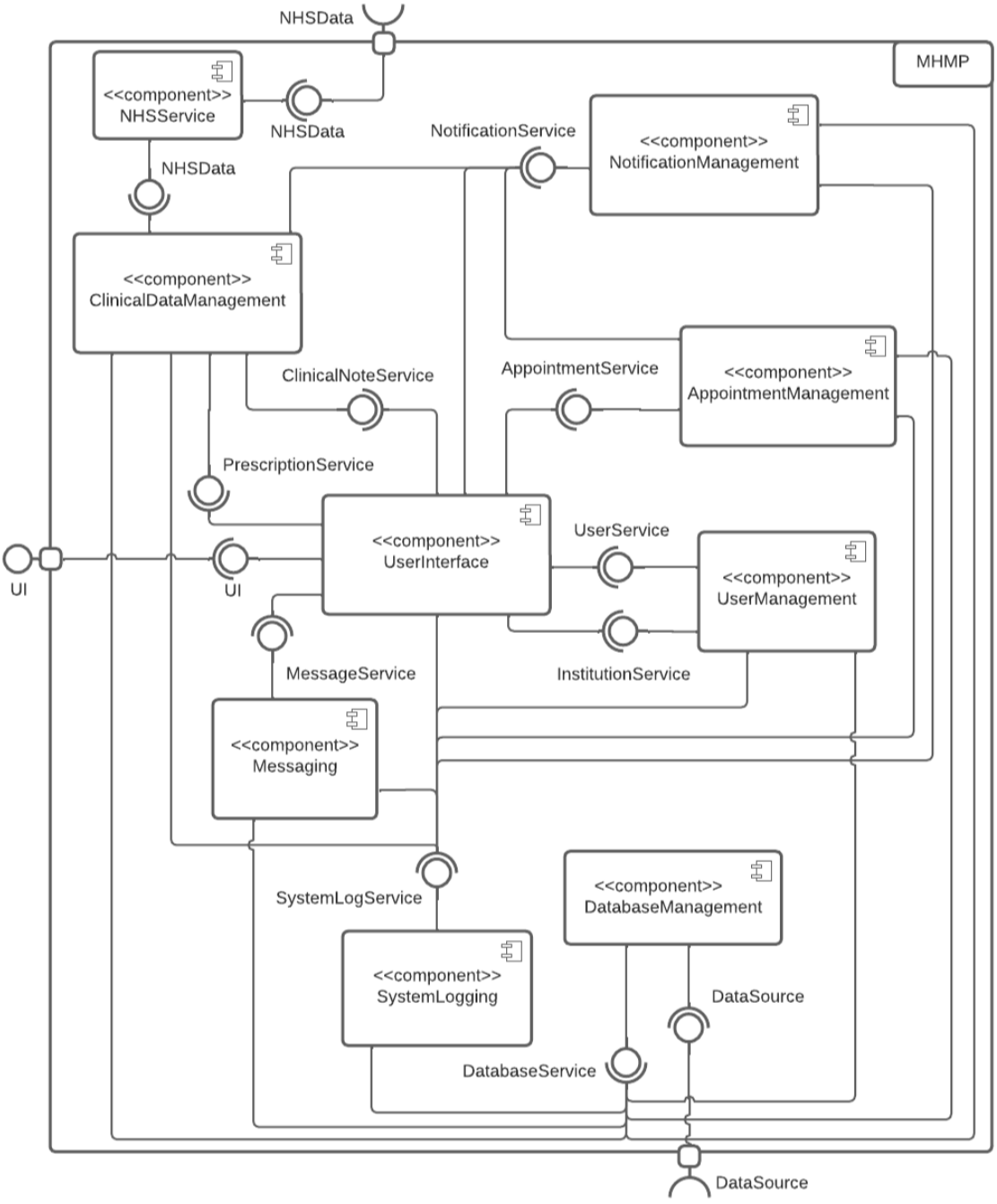


Figure 4.8.0.1: Component Diagram

For the above component diagram in **Figure 4.8.0.1** we have opted for the black-box style, see reference [24]. When modelling the system, it is best to use black-box views to focus on large-scale architectural concerns. Black-box style is good at showing a high level view of the key components in the system and how they are connected. In our case, it offers a contrasting view to the detailed design class diagram (which is closer to a white-box view).

4.9 Deployment Diagram

Deployment diagrams show where software is applied on hardware. The diagram demonstrates how different parts of the software are deployed, and how they connect to each other [14].

The deployment diagram (**Figure 4.9.0.1**) illustrates a multi-tiered architecture where different components of a system are deployed across various devices and servers, connected through specific communication protocols. The front-end tier consists of two types of devices: a PC/laptop and an iOS/Android mobile device. On the PC/laptop, a web app runs within a browser execution environment whilst in iOS and Android devices the app runs on a Javascript Engine (Hermes). There is an Application Server, which is an Azure Linux Virtual Machine, that hosts the Node.js runtime environment, serving as the execution environment for PC/laptop users. The back-end tier consists of an Azure Linux Virtual Machine hosting a PostgreSQL server. Communication between the tiers is facilitated by specific protocols. HTTPS is used for communication between the Browser and the Nginx web server. TCP/IP is used for communication between the database server and the Node.js server and between the database server and Hermes.

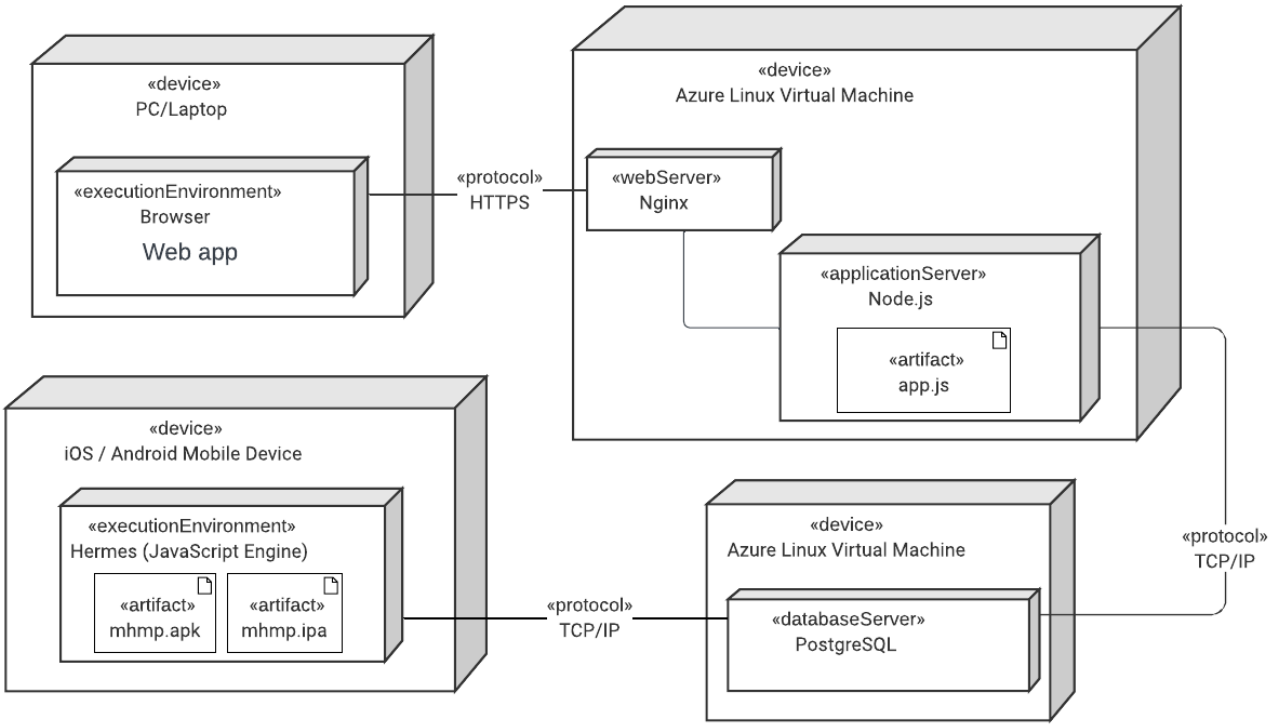


Figure 4.9.0.1: Deployment Diagram

Chapter 5

Application Mock-up

5.1 Overview

Presented in a desktop web application format, the mock-up screenshots in this chapter aim to showcase the application's design, layout, and functionality. Given our users may have lower levels of technological literacy and the busy schedules of professionals working in mental healthcare, we have aimed to keep the user interface simple, minimal, and usable to deliver an optimal user experience in accordance with one of Nielsen's 'Usability heuristics for interaction design': 'Aesthetic and Minimalist Design' [25] [26].

The following use cases for the application's core features have been included in our mock-ups:

- UC6: ViewClinicalNotes
- UC8: ManageAppointment
- UC11: RequestToRescheduleAppointment
- UC13: ReceiveAppointmentNotification
- UC16: ViewPrescription
- UC21: SendMessages

For all mock-up images shown, the top right of the screen has a home button, a profile / edit details button, a notifications button, and a messages button (patients do not have access to messaging, see **Figure 3.4.0.1**). These are highlighted when one of these sections is selected. These are using icons resembling relatable imagery in accordance with another one of Nielsen's heuristics: 'Match between the system and the real world', which suggests that designers should match user interface elements to real life objects [25].

Please see the Appendix, Section 8.3 for a more comprehensive selection of mock-up screenshots and a mock-up of some interfaces for the mobile app version of the MHMP.

5.2 Mock-Up

5.2.1 UC6: ViewClinicalNotes

After logging into the MHMP, a user navigates to the 'Select Patient' page from the main menu. Here, they are able to apply filters (Date-Of-Birth (DOB), Mental Health Act Status, and Clinical Status) and search for a patient by their name or NHS number. Should the search return the desired patient, they can select the patient and view their clinical notes. **Figure 5.2.1.2** shows the window to view psychiatric notes. In accordance with the read / write privileges outlined in **Table 2.2**, the psychiatrist is able to view patient details, psychiatric, therapist, and GP notes, appointments, the patient's prescriptions and general observations (General Obs in the user interface) in this section using the corresponding tabs. See the Appendix for the prescription / patient details tab mock-ups.

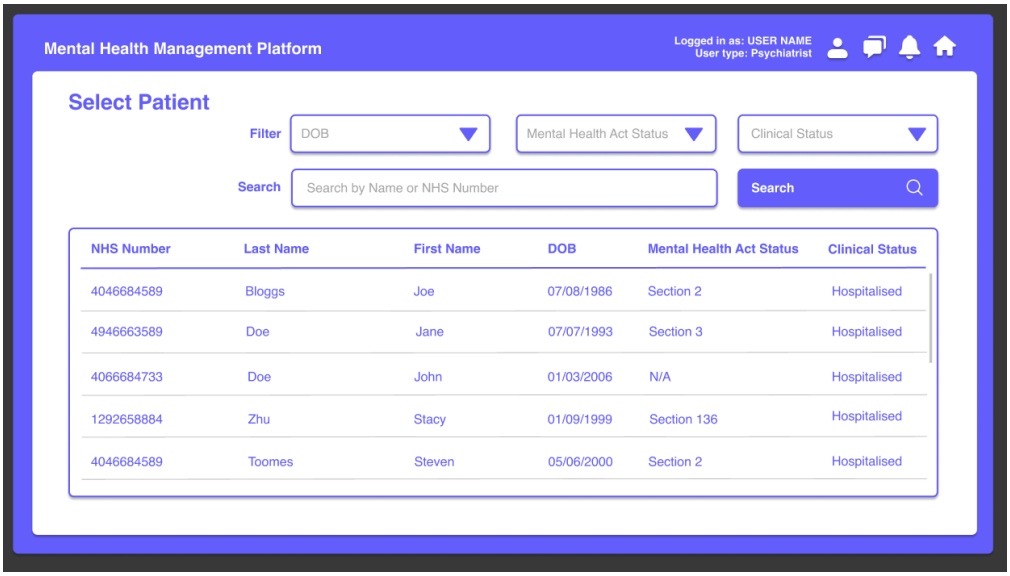


Figure 5.2.1.1: Select Patient Page (User type: Psychiatrist)

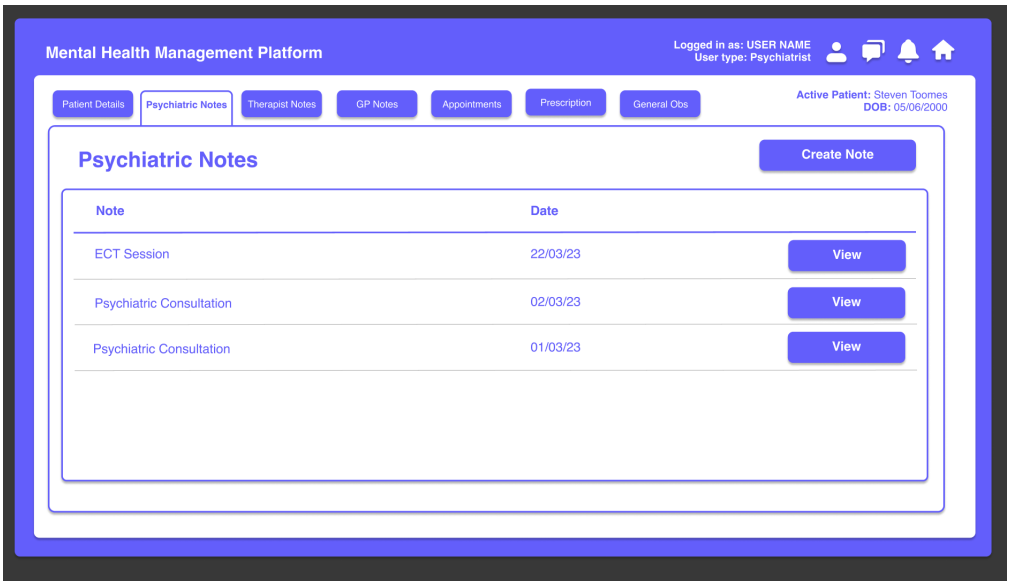


Figure 5.2.1.2: Patient Psychiatric Note Page (User type: Psychiatrist)

5.2.2 UC8: ManageAppointment

From the main menu, an individual with appointment management abilities (for example, a psychiatrist) navigates to the 'Your Appointments' page to access the in-app calendar (this section is named 'Appointment Manager' for Secretarial Professionals and System Admin main menu because they do not have appointments themselves - see Appendix). Here, they are able to view their scheduled appointments. They click 'Edit' to edit an appointment and an 'Edit Appointment' pop-up window is displayed, where they may cancel or update the appointment with the relevant buttons.

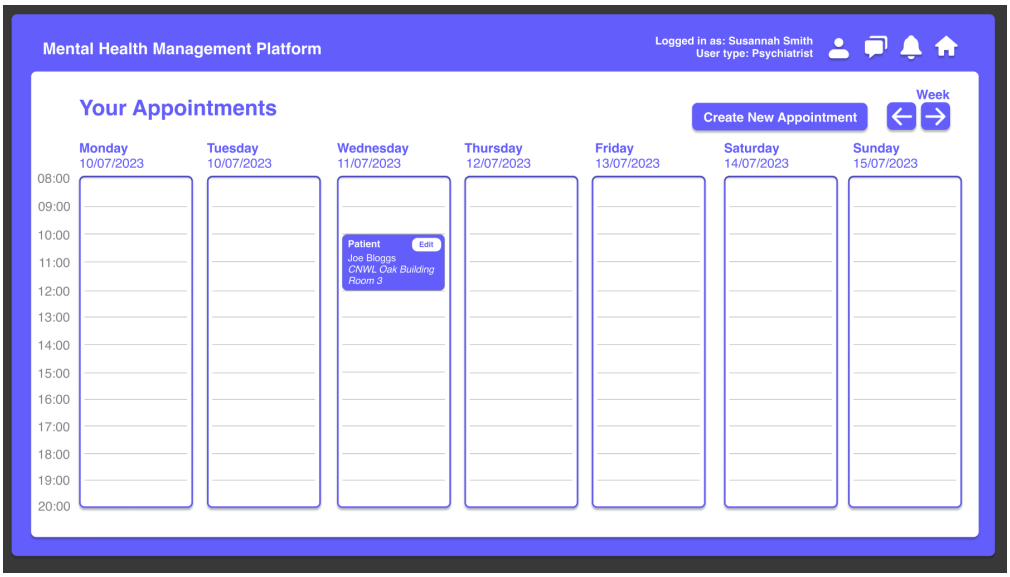


Figure 5.2.2.1: Your Appointments (Appointment Manager) Page (User type: Psychiatrist)

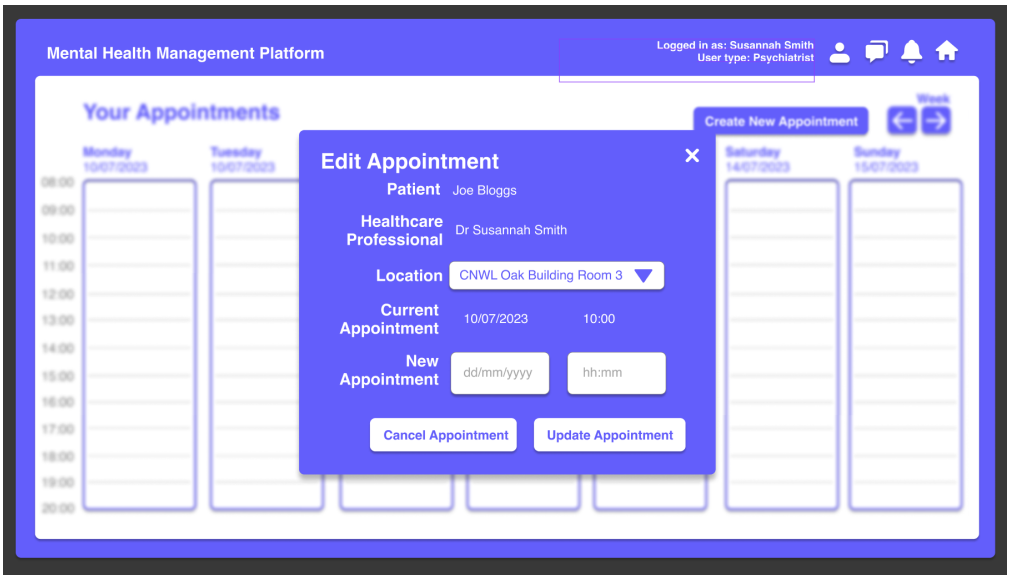


Figure 5.2.2.2: Edit Appointment Window (User type: Psychiatrist)

5.2.3 UC11: RequestToRescheduleAppointment

A patient navigates to the 'Your Appointments' page to access the in-app calendar. Here, they are able to view their scheduled appointments. They can select the Reschedule / Cancel button to carry out these actions for their chosen appointment, which will bring up a 'Reschedule/Cancel Appointment' pop-up window. They may propose a new appointment time by submitting a reschedule request for a secretarial professional to review. Please note that the patient user interface does not display a messaging tab because they cannot send or receive messages.

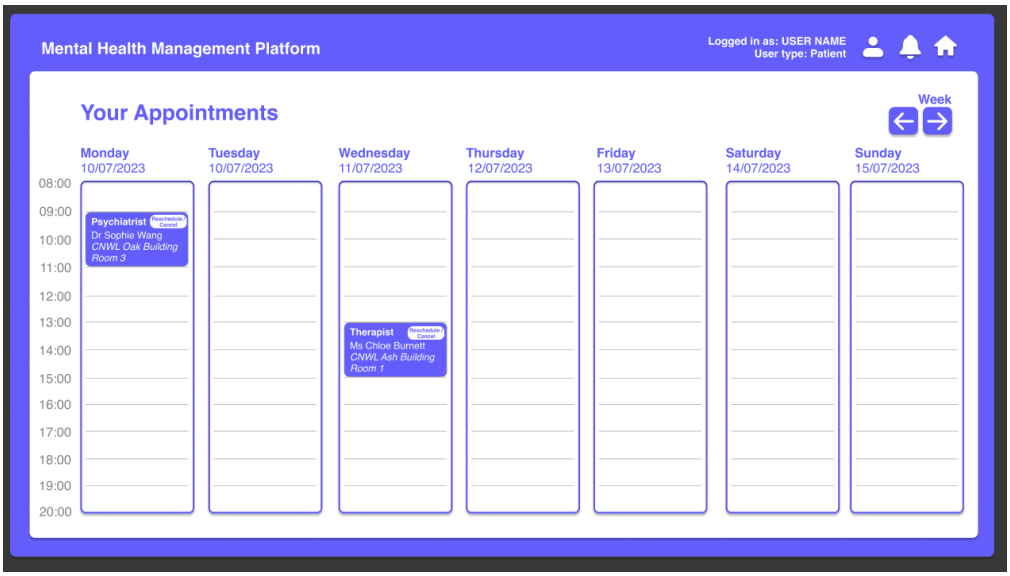


Figure 5.2.3.1: Your Appointments Page (User type: Patient)

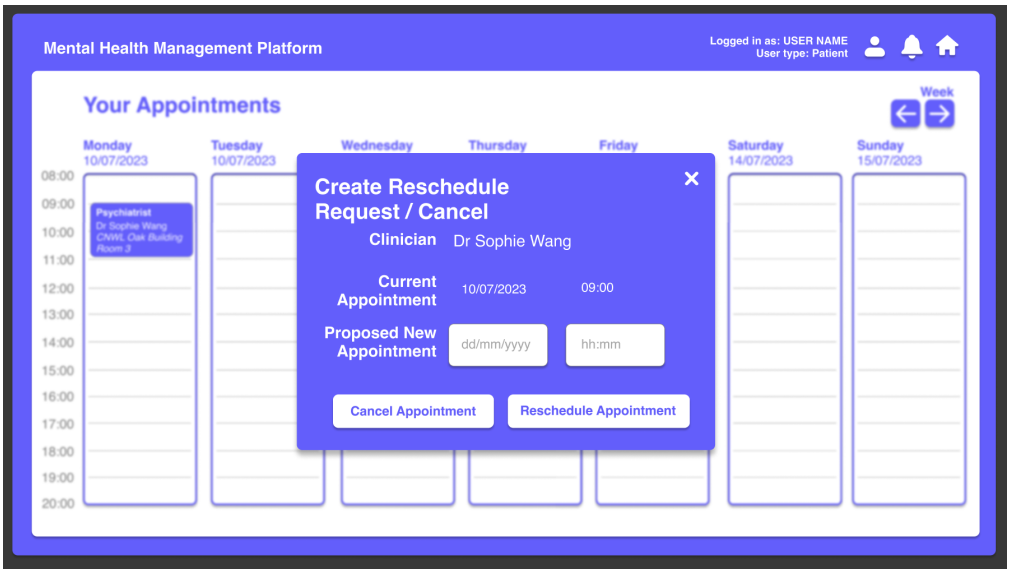


Figure 5.2.3.2: Reschedule/Cancel Appointment Window (User type: Patient)

5.2.4 UC13: ReceiveAppointmentNotification

A user (in this case, a patient) navigates to the 'Your Notifications' page from the top navigation bar. Here, they are able to view a list of notifications that they have received. Please note that the type of notification is displayed for the psychiatrist. Clicking on a specific notification will display a 'Notification' pop-up window that provides the full message associated with that notification; the details of a rescheduled appointment in this case.

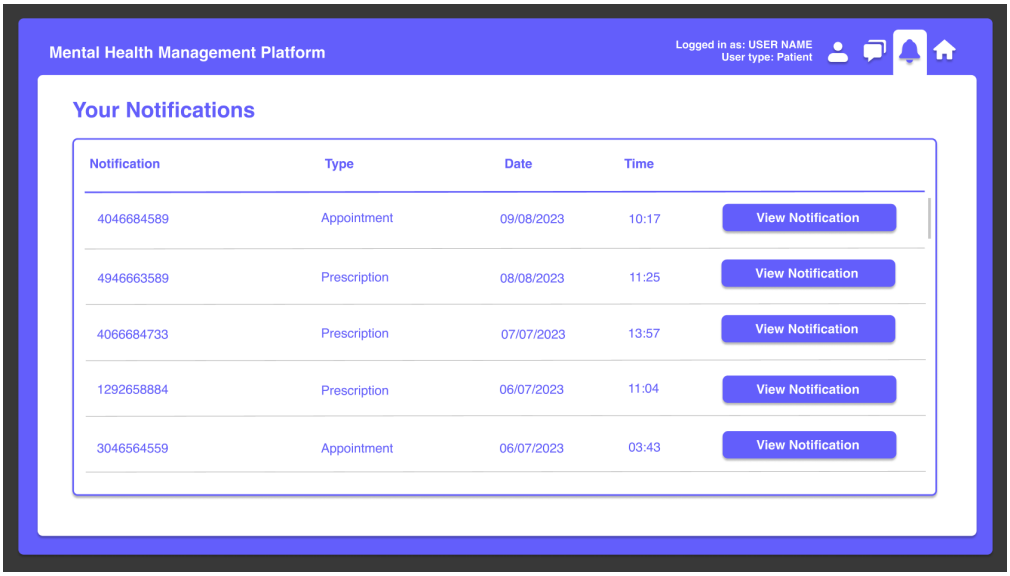


Figure 5.2.4.1: Your Notifications Page (User type: Patient)

Figure 5.2.4.2: Notification Window (User type: Patient)

5.2.5 UC16: ViewPrescription

From the main menu, a patient navigates to the 'Your Prescriptions' page. Here, they can view a list of prescriptions issued to them. Details including dosage and the issued date are provided for each prescription, as well as whether the chosen prescription is a repeat prescription. Additionally, there is an option for them to apply a filter to display only their repeat prescriptions.

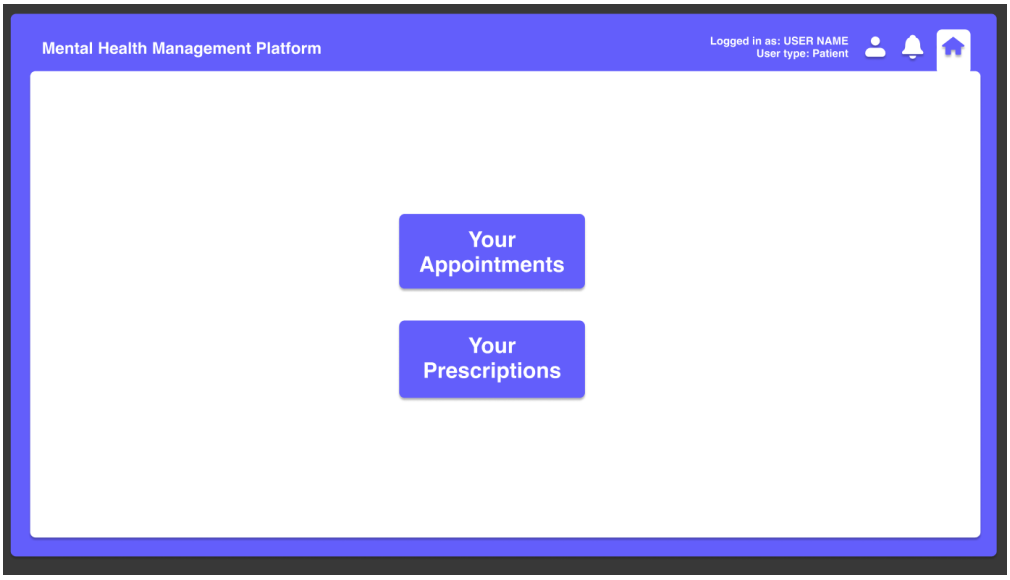


Figure 5.2.5.1: Main Menu Page (User type: Patient)

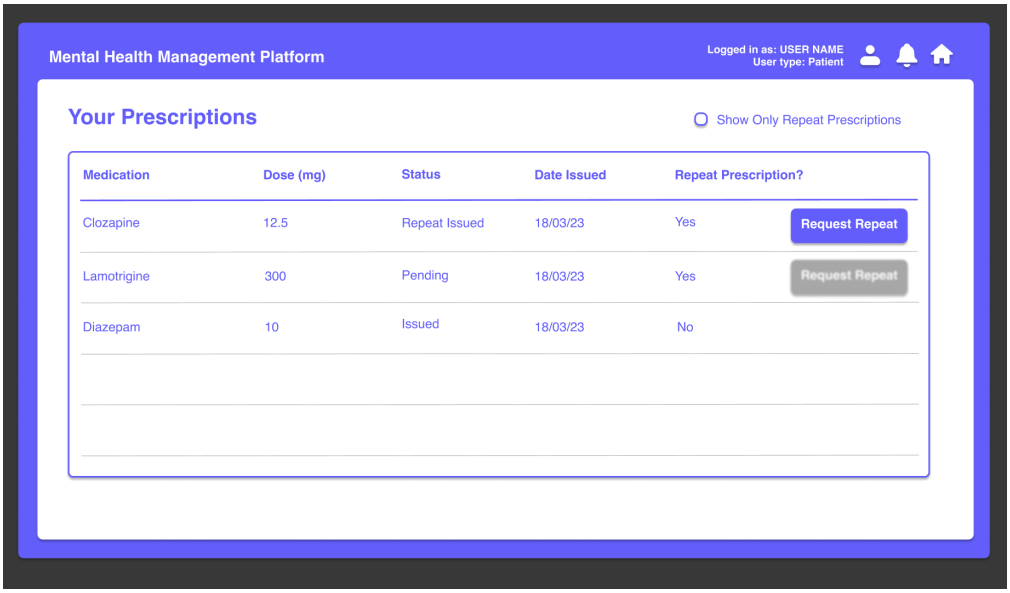


Figure 5.2.5.2: Your Prescriptions Page (User type: Patient)

5.2.6 UC21: SendMessages

A healthcare professional or a system administrator navigates to the 'Your Messages' page from the top navigation bar. The page lists existing chats. They have the option to select an existing chat or start a new chat. Selecting or starting a chat prompts a pop-up window to appear, where they can compose and send messages to the intended recipient. Additionally, we have created a version of this functionality for our mobile application to demonstrate what this key feature would look like on a mobile device - please see the Appendix.

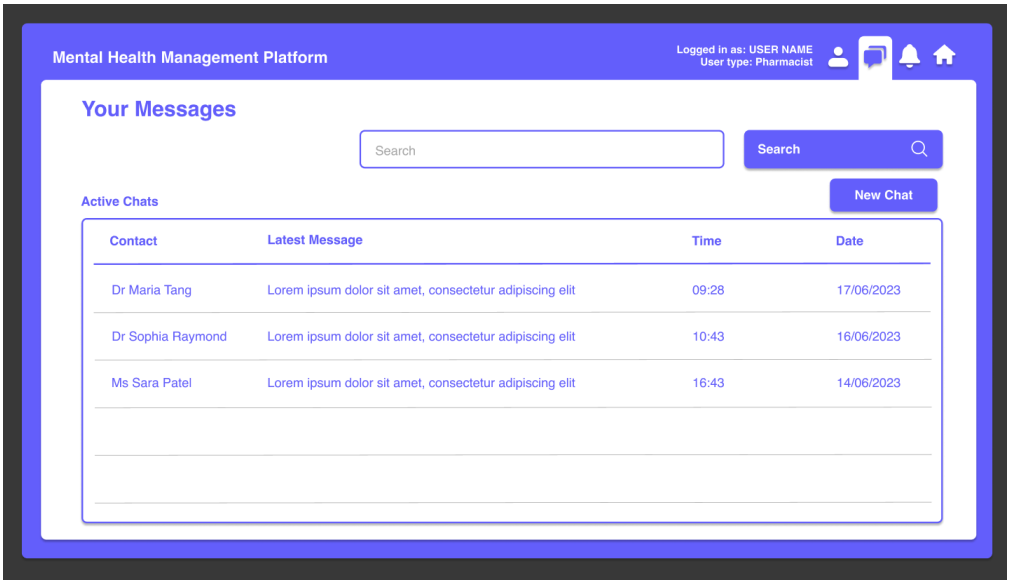


Figure 5.2.6.1: Your Messages Page (User type: Pharmacist)

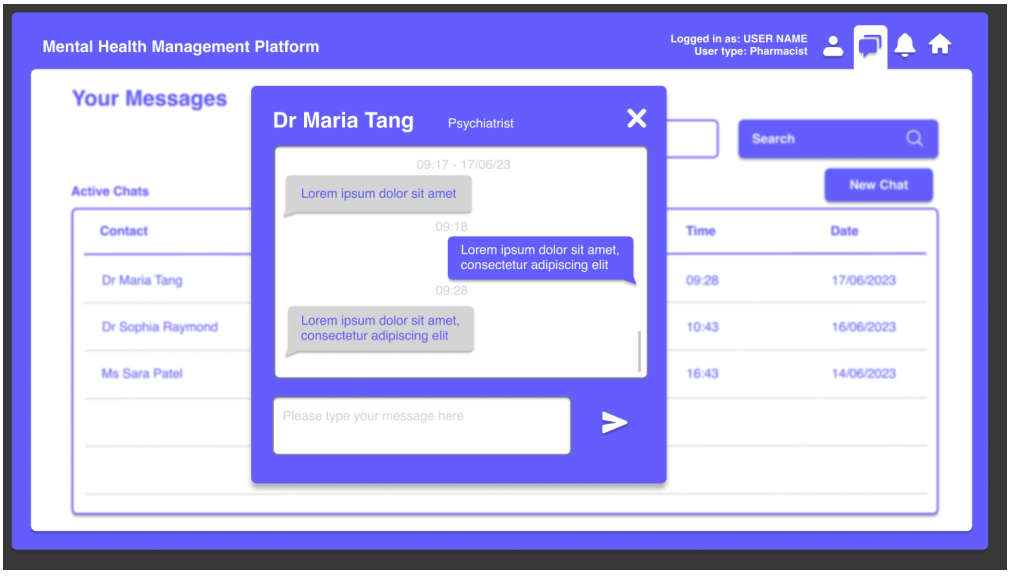


Figure 5.2.6.2: Chat Window (User type: Pharmacist

Chapter 6

Summary and Conclusions

6.1 Summary

In this report, we have presented a comprehensive overview of the Mental Health Management Platform (MHMP), a proposed web application designed to enhance the delivery and coordination of mental health services in the UK. The MHMP aims to bridge the gap between mental health professionals and patients by providing a user-friendly, efficient, and interconnected platform for communication and collaboration.

We began by outlining the project brief, background, and purpose of the MHMP, highlighting its key features and comparing it to existing applications in the domain. We then detailed the functional and non-functional requirements of the MHMP system using the MoSCoW prioritisation method and developed a domain model to represent the main concepts and relationships within the system. Using the requirements as a guide, we identified and specified the use cases for the MHMP, detailing the interactions between the system and its actors. We presented a UML use case diagram to provide visualisation of these interactions and provided a requirements traceability matrix to map the use cases to their corresponding requirements.

An extensive domain analysis was undertaken to identify and categorise system entities, actors, and their interrelations, laying the groundwork for the object-oriented framework. We made sure to carry out a meticulous object identification process to ensure a coherent, synchronised system behaviour and a design that is scalable, maintainable, and aligned with software engineering best practices for complex system development. The analysis class diagram was initially crafted to represent the problem domain, serving as a visual depiction of its key elements. This diagram subsequently formed the basis for creating the design class diagram, which detailed the specifics of implementation. The process of creating analysis and design class UML diagrams delineated the system's static structure.

These diagrams were used as a foundation for sequence diagrams for modelling object interactions over time, state machine diagrams capturing object lifecycles, activity diagrams to illustrate activity flows for different business cases, and component and deployment diagrams for detailing system organisation and deployment architecture. These diagrams collectively describe the MHMP's static and dynamic architecture.

Finally, we presented an application mock-up showcasing the MHMP's design, layout, and functionality for several core use cases. The mock-ups demonstrate our commitment to delivering a simple, usable, and optimal user experience for the diverse parties involved in mental health care within the UK.

6.2 Conclusions

In conclusion, the MHMP is a promising solution to address the critical need for integrated communication and collaboration within the UK's mental health sector. By leveraging technology, the MHMP aims to streamline operational processes and improve the mental health care experience for patients and professionals.

The comprehensive analysis, design, and mock-ups presented in this report lay a solid foundation for the future development and implementation of the MHMP. If the project were to move forward into the implementation stage, it would be crucial to engage with stakeholders, incorporate their feedback, and ensure the system meets the evolving needs of all those involved in providing and receiving mental health care.

Chapter 7

Task Distribution

Key: Leading or co-leading role = ✓ Supporting role = (✓)

Team Member:	**OA**	**JB**	**CH**	**AH**	**MP**	**KT**
Chapter 1						
Similar project analysis					✓	
Report writing	(✓)		(✓)		✓	
Chapter 2						
MoSCoW requirements	(✓)	✓	✓	✓	✓	(✓)
Domain model		(✓)		✓		
Report writing		✓	(✓)			✓
Chapter 3						
Use case diagram	(✓)	(✓)	(✓)	✓	(✓)	(✓)
Use case specifications		✓		✓		✓
Report writing						✓
Chapter 4						
Class diagram (analysis)		✓		✓		
Class diagram (design)		✓		✓		
Component diagram	(✓)	✓	(✓)	✓	(✓)	
Deployment diagram		✓	(✓)	✓	(✓)	
Activity diagrams		✓	✓	✓	✓	
Sequence diagrams		✓	✓	✓	(✓)	
State machine diagrams		✓	(✓)	✓		
Report writing		✓	✓	✓	✓	
Chapter 5						
Interactive mock-up					✓	
Demonstration video recording					✓	
Report writing					✓	✓
Chapter 6						
Report writing					✓	
Other						
Report set-up, formatting, management			✓			
Other report chapters (excluding 1-6)			✓		✓	
Video editing	✓	✓	(✓)	✓	(✓)	
Video content and recording	✓	✓	✓	✓	✓	✓

Note: Although not shown in the table, contributions were made in the diagram development, report refining and thorough checking process by all members.

Note: In this task table, the diagram tasks are represented as standalone tasks, and the explanations for them are covered in the corresponding *report writing* task.

Chapter 8

Appendix

8.1 Chapter 1 - Field Research Healthcare Professional Feedback

Here, we have paraphrased the key points from discussions with healthcare professionals working in the NHS who have worked or studied in the context of psychiatry to help inform the development of our application:

- Poor user interface of existing applications means important details can be missed.
- Inability to issue prescriptions from existing app-based solutions in psychiatry is frustrating.
- Frequent use of excel spreadsheets to manage information - because of the lack of automated reminders, important activities are missed.
- Dedicated electronic app solution would be helpful for psychiatry.
- No access to basic GP records in psychiatry is frustrating.
- Would be beneficial of having one system that was capable of a lot of things and used across platforms in the local area or nationally.
- Existing applications contain many superfluous features which are not necessary for psychiatry (for example, imaging is not necessary). A streamlined solution would be beneficial.
- Messaging is a key feature of certain existing applications and healthcare professionals use it frequently.

8.2 Chapter 4 - Object-Oriented Analysis and Design Models

8.2.1 PlantUML Code for OOD class diagram

In order to enhance development efficiency, we chose to use PlantUML. This tool allows for quick adjustments to our class diagrams, supporting a more efficient iterative process. PlantUML uses a plain text language to generate UML diagrams, making it easy to visualise complex system architectures.

Below is our complete design class diagram in PlantUML code. For visualisation, set up a local renderer as per instructions at `https://plantuml.com/starting`, or use an online visualizer like `https://www.planttext.com/`.

Listing 8.1: UML diagram code

```
@startuml OOD_New

skinparam class {
    BackgroundColor Snow
    BorderColor Black
    FontName Arial
    FontColor Black
    FontStyle Normal
    ArrowThickness 1
}
skinparam linetype ortho

class UI << boundary >>{
  - object: String
  - recordID: String
  + displayRecords(object: String, filter: Map<String, String>): void
  + displayRecord(recordID: String): void
  + updateRecord(record: Map<String, String>): void
  + deleteRecord(recordID: String): void
}

class DatabaseService << boundary >>{
  + createRecord(record: Map<String, String>): String {static}
  + updateRecord(record: Map<String, String>): boolean {static}
  + deleteRecord(id: String): boolean {static}
  + getRecords(query: String): List<Map<String, String>> {static}
}

class NHSService << boundary >>{
  + createRecord(record: Map<String, String>): bool{static}
  + getRecords(nhsNumber: String, type: String): List<Map<String, String>> {static}
  + requestRepeatPrescription(prescriptionID: String): String {static}
}

UI ..> UserService : << uses >>
UI ..> InstitutionService : << uses >>
UI ..> NotificationService : << uses >>
UI ..> ClinicalNoteService : << uses >>
UI ..> PrescriptionService : << uses >>
UI ..> MessageService : << uses >>
UI ..> AppointmentService : << uses >>
UI ..> SystemLogService : << uses >>
```

```
DatabaseService <.. UserService : << uses >>
DatabaseService <.. InstitutionService : << uses >>
DatabaseService <.. NotificationService : << uses >>
DatabaseService <.. ClinicalNoteService : << uses >>
DatabaseService <.. PrescriptionService : << uses >>
DatabaseService <.. MessageService : << uses >>
DatabaseService <.. AppointmentService : << uses >>
DatabaseService <.. SystemLogService : << uses >>

class User << entity >>{
  # userID: String
  # firstName: String
  # lastName: String
  # status: String
  # userType: String
  # emailAddress: String
  # password: String
  # createdBy: String
  # createdOn: DateTime
  # updatedBy: String
  # updatedOn: DateTime
  + User() <<constructor>>
  + getUserID(): String
  + getFirstName(): String
  + setFirstName(firstName: String): void
  + getLastName(): String
  + setLastName(lastName: String): void
  + getStatus(): String
  + setStatus(status: String): void
  + getUserType(): String
  + setUserType(userType: String): void
  + getEmailAddress(): String
  + setEmailAddress(emailAddress: String): void
  + getPassword(): String
  + setPassword(password: String): void
  + getCreatedBy(): String
  + setCreatedBy(createdBy: String): void
  + getCreatedOn(): DateTime
  + setCreatedOn(createdOn: DateTime): void
  + getUpdatedBy(): String
  + setUpdatedBy(updatedBy: String): void
  + getUpdatedOn(): DateTime
  + setUpdatedOn(updatedOn: DateTime): void
}

note left of User::userType
  Available values:
  patient,
  admin,
  psychiatrist,
  therapist,
  nurse,
  secretarial professional,
  pharmacist
end note

class Patient << entity >> extends User {
  - nhsNumber: String
  - mentalHealthActStatus: String
  - clinicalStatus: String
  - comments: String
  - dateOfBirth: DateTime
  + Patient() <<constructor>>
  + getNhsNumber(): String
  + setNhsNumber(nhsNumber: String): void
  + getMentalHealthActStatus(): String
  + setMentalHealthActStatus(mentalHealthActStatus: String): void
  + getClinicalStatus(): String
  + setClinicalStatus(clinicalStatus: String): void
```

```
  + getComments(): String
  + setComments(comments: String): void
  + getDateOfBirth(): DateTime
  + setDateOfBirth(dateOfBirth: DateTime): void
}

class HealthcareProfessional << entity >> extends User {
  - institutionID: String
  + HealthcareProfessional() <<constructor>>
  + getInstitutionID(): String
  + setInstitutionID(institutionID: String): void
}

class Institution << entity >> {
  - institutionID: String
  - institutionName: String
  - institutionAddress: String
  - institutionPostcode: String
  + Institution() <<constructor>>
  + getInstitutionID(): String
  + getInstitutionName(): String
  + setInstitutionName(institutionName: String): void
  + getInstitutionAddress(): String
  + setInstitutionAddress(institutionAddress: String): void
  + getInstitutionPostcode(): String
  + setInstitutionPostcode(institutionPostcode: String): void
}

class UserService << control >> {
  + createPatient(emailAddress:String, nhsNumber: String): Patient {static}
  + createHealthcareProfessional(emailAddress:String, nhsNumber: String): HealthcareProfessional {static}
  + createSystemAdmin(emailAddress:String): User {static}
  + getAdmins(id: String, name: String, {static}
  emailAddress: String, createdOn: DateTime): List<User> {static}
  + getPatients(id: String, name: String, emailAddress: String, createdOn: DateTime, {static}
  nhsNumber: String, mentalHealthActStatus: String, clinicalStatus: String, dateOfBirth: DateTime): List<Patient>
      {static}
  + getHealthcareProfessionals(id: String, name: String, emailAddress: String, {static}
  createdOn: DateTime, institutionID: String): List<HealthcareProfessional> {static}
  + updatePatient(patient: Patient, newValues: Map<String, String>): Patient {static}
  + updateHealthcareProfessional(healthcareProfessional: HealthcareProfessional, newValues: Map<String, String>):
      HealthcareProfessional {static}
  + updateAdmin(admin: User, newValues: Map<String, String>): User {static}
  + deleteUser(id: String): boolean {static}
  + login(emailAddress: String, password: String): boolean {static}
  + logout(id: String): boolean {static}
  + inSameInstitution(userID1: String, userID2: String): boolean {static}
  ' TBD useful in sequence diagram
  + getPatientID(nhsNumber: String): String {static}
}

class InstitutionService << control >> {
  + createInstitution(institutionName: String, institutionAddress: String, {static}
  institutionPostcode: String): Institution {static}
  + getInstitution(id: String, name: String, postCode: String): List<Institution> {static}
  + updateInstitution(institution: Institution, newValues: Map<String, String>): Institution {static}
  + deleteInstitution(id: String): boolean {static}
}

class AppointmentService << control >> {
  + createAppointment(patientID: String, healthcareProfessionalID: String, {static}
  type: String, dateTime: DateTime, location: String): Appointment {static}
  + getAppointments(appointmentID: String, patientID: String, healthcareProfessionalID: String, {static}
  dateTime: DateTime, type: String, location: String, status: String): List<Appointment> {static}
  + updateAppointment(appointment: Appointment, newValues: Map<String, String>): Appointment {static}
  + cancelAppointment(id: String): boolean {static}
  + deleteAppointment(id: String): boolean {static}
  + createRequest(appointmentID: String, requestedBy: String, proposedTime: DateTime):
      AppointmentRescheduleRequest {static}
```

```
  + getRequests(appointmentID: String, createdBy: String, status: String, {static}
  updatedBy: String): List<AppointmentRescheduleRequest> {static}
  + updateRequest(request: AppointmentRescheduleRequest, newValues: Map<String, String>):
      AppointmentRescheduleRequest {static}
  + deleteRequest(id: String): boolean {static}
}

class AppointmentRescheduleRequest << entity >> {
  - requestID: String
  - appointmentID: String
  - createdOn: DateTime
  - createdBy: String
  - updatedBy: String
  - updatedOn: DateTime
  - proposedTime: DateTime
  - status: String
  + AppointmentRescheduleRequest() <<constructor>>
  + getRequestID(): String
  + getAppointmentID(): String
  + setAppointmentID(appointmentID: String): void
  + getCreatedOn(): DateTime
  + setCreatedOn(createdOn: DateTime): void
  + getCreatedBy(): String
  + setCreatedBy(createdBy: String): void
  + getUpdatedBy(): String
  + setUpdatedBy(updatedBy: String): void
  + getUpdatedOn(): DateTime
  + setUpdatedOn(updatedOn: DateTime): void
  + getProposedTime(): DateTime
  + setProposedTime(proposedTime: DateTime): void
  + getStatus(): String
  + setStatus(status: String): void
}

class Appointment << entity >> {
  - appointmentID: String
  - type: String
  - location: String
  - dateTime: DateTime
  - status: String
  - patientID : String
  - healthcareProfessionalID: String
  - createdBy: DateTime
  - createdOn: DateTime
  - updatedBy: String
  - updatedOn: DateTime
  + Appointment() <<constructor>>
  + getAppointmentID(): String
  + getType(): String
  + setType(type: String): void
  + getLocation(): String
  + setLocation(location: String): void
  + getDateTime(): DateTime
  + setDateTime(dateTime: DateTime): void
  + getStatus(): String
  + setStatus(status: String): void
  + getPatientID(): String
  + setPatientID(patientID: String): void
  + getHealthcareProfessionalID(): String
  + setHealthcareProfessionalID(healthcareProfessionalID: String): void
  + getCreatedBy(): DateTime
  + setCreatedBy(createdBy: DateTime): void
  + getCreatedOn(): DateTime
  + setCreatedOn(createdOn: DateTime): void
  + getUpdatedBy(): String
  + setUpdatedBy(updatedBy: String): void
  + getUpdatedOn(): DateTime
  + setUpdatedOn(updatedOn: DateTime): void
}
```

```
class ClinicalNote << entity >> {
  - noteID: String
  - type: String
  - patientNhsNumber: String
  - content: String
  - createdBy: String
  - createdOn: DateTime
  - updatedBy: String
  - updatedOn: DateTime
  + ClinicalNote() <<constructor>>
  + getNoteID(): String
  + getNoteType(): String
  + setNoteType(noteType: String): void
  + getPatientNhsNumber(): String
  + setPatientNhsNumber(patientNhsNumber: String): void
  + getNoteContent(): String
  + setNoteContent(noteContent: String): void
  + getCreatedBy(): String
  + setCreatedBy(createdBy: String): void
  + getCreatedOn(): DateTime
  + setCreatedOn(createdOn: DateTime): void
  + getUpdatedBy(): String
  + setUpdatedBy(updatedBy: String): void
  + getUpdatedOn(): DateTime
  + setUpdatedOn(updatedOn: DateTime): void
}

class ClinicalNoteService << control >>{
  + createClinicalNote(patientNhsNumber: String, type: String, content: String): ClinicalNote {static}
  + getNotes(id: String, type: String, patientNhsNumber: String, {static}
  createdBy: String, updatedBy: String): List<ClinicalNote> {static}
  + updateNote(note: ClinicalNote, newValues: Map<String, String>): ClinicalNote {static}
  + deleteNote(id: String): boolean {static}
}

class Prescription << entity >> {
  - prescriptionID: String
  - patientNhsNumber: String
  - createdBy: String
  - createdOn: DateTime
  - issuedBy: String
  - issuedOn: DateTime
  - description: String
  - repeatable: boolean
  - status: String
  + Prescription() <<constructor>>
  + getPrescriptionID(): String
  + getPatientNhsNumber(): String
  + setPatientNhsNumber(patientNhsNumber: String): void
  + getCreatedBy(): String
  + setCreatedBy(createdBy: String): void
  + getCreatedOn(): DateTime
  + setCreatedOn(createdOn: DateTime): void
  + getDescription(): String
  + setDescription(description: String): void
  + getStatus(): String
  + setStatus(status: String): void
}

note right of Prescription::status
  Available values: Flagged,
  Reviewed,
  Issued,
  Pending
  Rejected,
  Repeat Issued,
  Repeat Rejected
end note
```

```
class PrescriptionService << control >> {
  + createPrescription(patientNhsNumber: String, createdBy: String, description: String): Prescription {static}
  + getPrescriptions(id: String, patientNhsNumber: String, createdBy: String, createdOn: DateTime, {static}
  flaggedStatus: String, isIssued: boolean): List<Prescription> {static}
  + updatePrescription(prescription: Prescription, newValues: Map<String, String>): Prescription {static}
  + deletePrescription(id: String): boolean {static}
  + requestRepeatPrescription(id: String): void {static}
  + scheduledGetPrescriptions(): void {static}
}

class Notification{
  - notificationID: String
  - userID: String
  - title: String
  - body: String
  - type: String
  - createdOn: DateTime
  + Notification() <<constructor>>
  + getNotificationID(): String
  + getUserID(): String
  + setUserID(userID: String): void
  + getCreatedOn(): DateTime
  + setCreatedOn(createdOn: DateTime): void
  + getType(): String
  + setType(type: String): void
  + getBody(): String
  + setBody(body: String): void
  + getTitle(): String
  + setTitle(title: String): void
}

class NotificationService << control >>{
  + createNotification(userID: String, title: String, {static}
  body: String, type: String): Notification {static}
  + getNotifications(id: String, userID: String, type: String, {static}
  createdOn: DateTime, read: boolean): List<Notification> {static}
  + updateNotification(notification: Notification, read: boolean): Notification {static}
  + deleteNotification(notificationID: String): boolean {static}
}

class Message << entity >> {
  - messageID: String
  - senderID: String
  - receiverID: String
  - content: String
  - createdOn: DateTime
  - read: boolean
  + Message() <<constructor>>
  + getMessageID(): String
  + getSenderID(): String
  + setSenderID(senderID: String): void
  + getReceiverID(): String
  + setReceiverID(receiverID: String): void
  + getContent(): String
  + setContent(content: String): void
  + getCreatedOn(): DateTime
  + getStatus(): boolean
  + setStatus(status: boolean): void
}

class MessageService << control >>{
  + createMessage(senderID: String, receiverID: String, content: String): Message {static}
  + getMessages(id: String, senderID: String, receiverID: String, {static}
  createdOn: DateTime, status: boolean): List<Message> {static}
  + updateMessage(message: Message, newValues: Map<String, String>): Message {static}
  + deleteMessage(messageID: String): boolean {static}
}
```

```
class SystemLog << entity >> {
  - logID: String
  - userID: String
  - action: String
  - oldValue: String
  - newValue: String
  - field: String
  - recordID: String
  - createdOn: DateTime
  - note: String
  + SystemLog() <<constructor>>
  + getLogID(): String
  + getUserID(): String
  ' + setUserID(userID: String): void
  + getAction(): String
  ' + setAction(action: String): void
  + getCreatedOn(): DateTime
  ' + setCreatedOn(createdOn: DateTime): void
  + getNote(): String
  ' + setNote(note: String): void
  + getOldValue(): String
  ' + setOldValue(value: String): void
  + getNewValue(): String
  ' + setNewValue(value: String): void
  + getField(): String
  ' + setField(field: String): void
}

note left of SystemLog
  Logs are read-only and cannot be updated.
  Values are set upon creation.
end note

class SystemLogService << control >>{
  + createLog(userID: String, action: String, oldValue: String, newValue: String, {static}
  field: String, recordID: String, note: String): boolean {static}
  + getLogs(logID: String, userID: String, action: String, oldValue : String, {static}
  newValue: String, field: String, recordID: String, createdOn: DateTime, note: String): List<SystemLog> {static}
  + deleteLog(logID: String): boolean {static}
}

note left of SystemLogService::deleteLog
  This method is only used by scheduled tasks to delete logs older than a certain date
end note

' UserService
UserService ..> Patient : << uses >>
UserService ..> HealthcareProfessional : << uses >>

' AppointmentService
AppointmentService ..> Appointment : << uses >>
AppointmentService ..> AppointmentRescheduleRequest : << uses >>

' Others
Institution "1" --> "1..*" HealthcareProfessional : has

InstitutionService ..> Institution : << uses >>

ClinicalNote <.. ClinicalNoteService : << uses >>
ClinicalNoteService ..> NHSService : << uses >>

PrescriptionService ..> Prescription : << uses >>
PrescriptionService ..> NHSService : << uses >>
SystemLogService ..> SystemLog : << uses >>

UserService ..> SystemLogService : << uses >>
PrescriptionService ..> SystemLogService : << uses >>
ClinicalNoteService ..> SystemLogService : << uses >>
AppointmentService ..> SystemLogService : << uses >>
```

```
InstitutionService ..> SystemLogService : << uses >>

AppointmentService ..> NotificationService : << uses >>
PrescriptionService ..> NotificationService : << uses >>
NotificationService ..> Notification : << uses >>
NotificationService ..> SystemLogService : << uses >>

MessageService ..> Message : << uses >>
MessageService ..> SystemLogService : << uses >>

PrescriptionService ..> UserService : << uses >>
```

8.2.2 List Values (Enumerations) for Attributes

- status
  - Appointment: [“Scheduled”, “Completed”, “Cancelled”,]
  - AppointmentRescheduleRequest: [“Pending”, “Resolved”]
  - Prescription: [“Issued”, “Rejected”, “Flagged”, “Reviewed”, “Pending”, “Repeat Issued”, “Repeat Rejected”]
  - User: [“Active”, “Inactive”]
- type
  - Appointment: [“Psychiatrist”, “Therapist”, “Time blocker”]
  - ClinicalNote: [“GP Note”, “Psychiatric Note”, “Therapist Note”, “General Observation”]
  - Notification: [“Appointment”, “Prescription”]
  - User: [“Patient”, “Psychiatrist”, “Therapist”, “Nurse”, “Secretarial Professional”, “Pharmacist”, “Admin”]
- clinicalStatus
  - Patient: [“Hospitalised”, “Intensive community support”, “General community support”, “Not receiving support”]

8.2.3 Full OOA class diagram

For completeness, we have included the full object-oriented analysis class diagram. Please see **Figure 8.2.3.1**

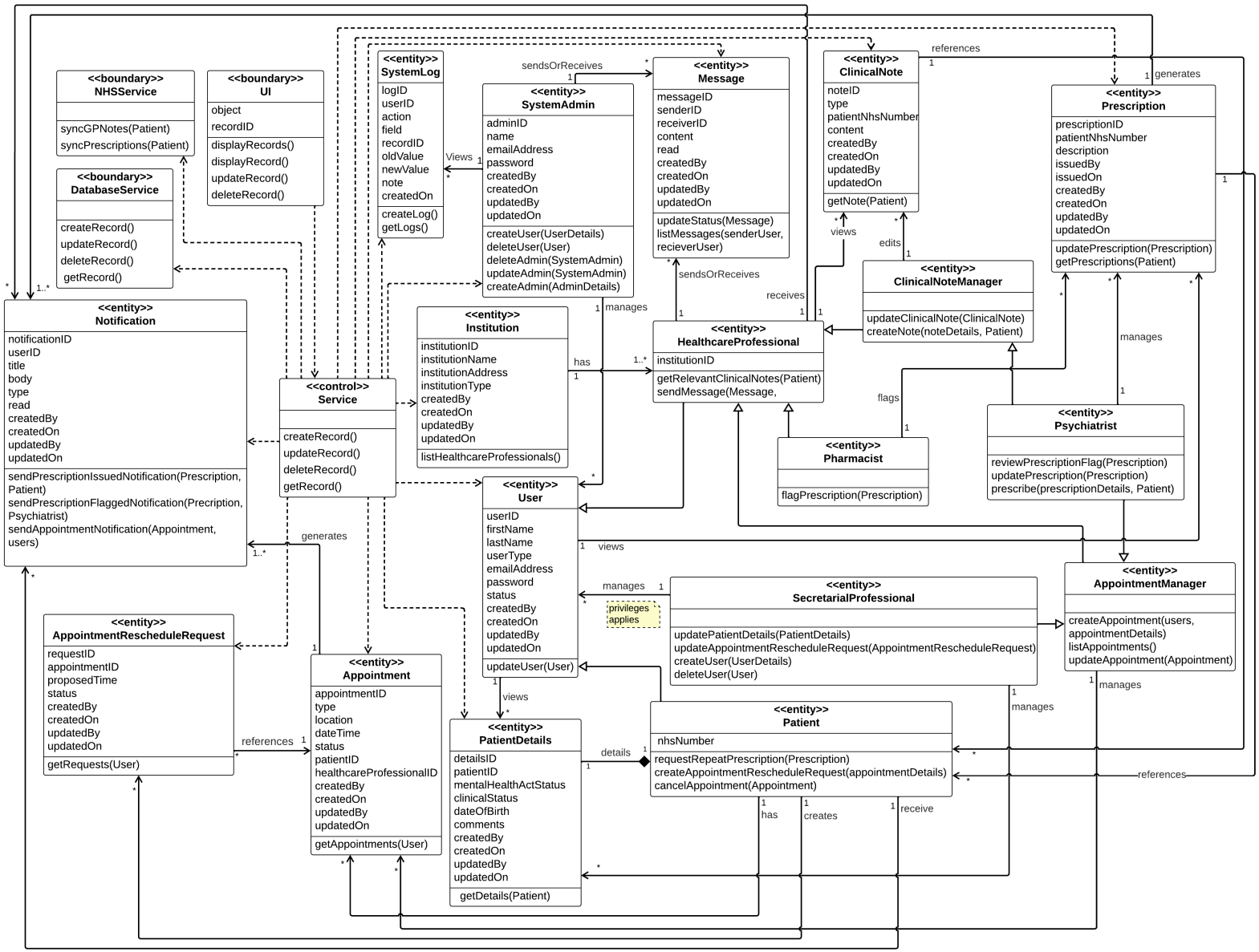


Figure 8.2.3.1: Analysis Class Diagram - Full Version

8.3 Chapter 5 - Mock Up - Extra Views

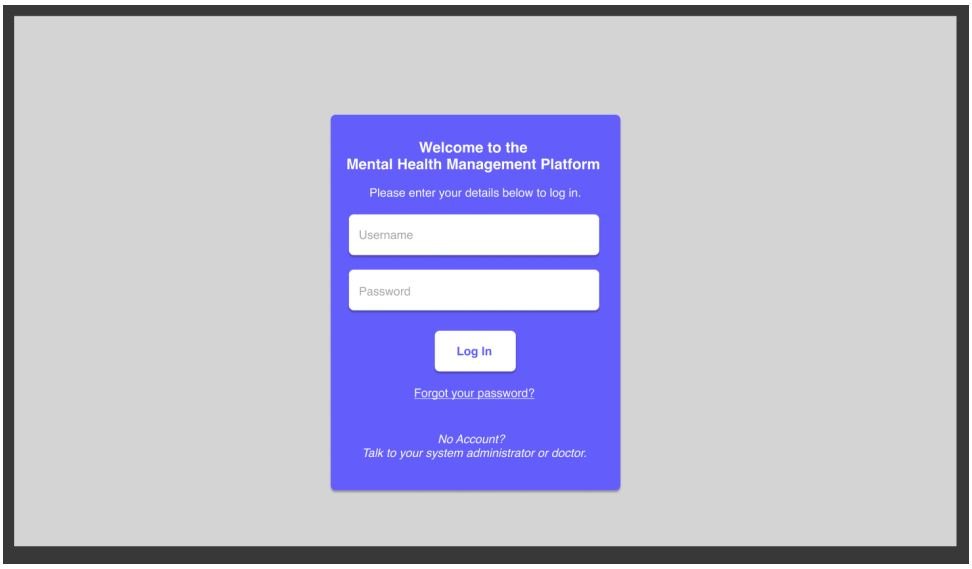


Figure 8.3.0.1: Login Page

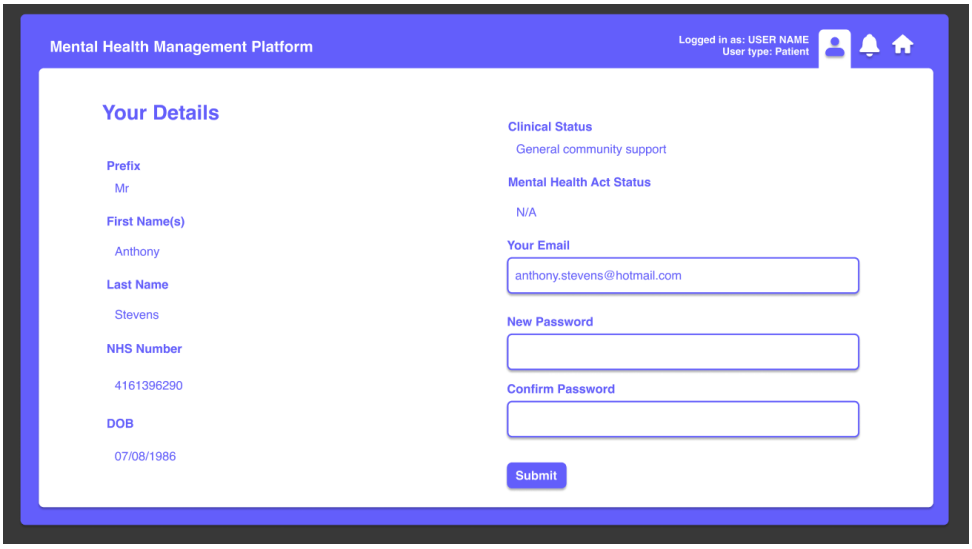


Figure 8.3.0.2: Edit Details - Patient View

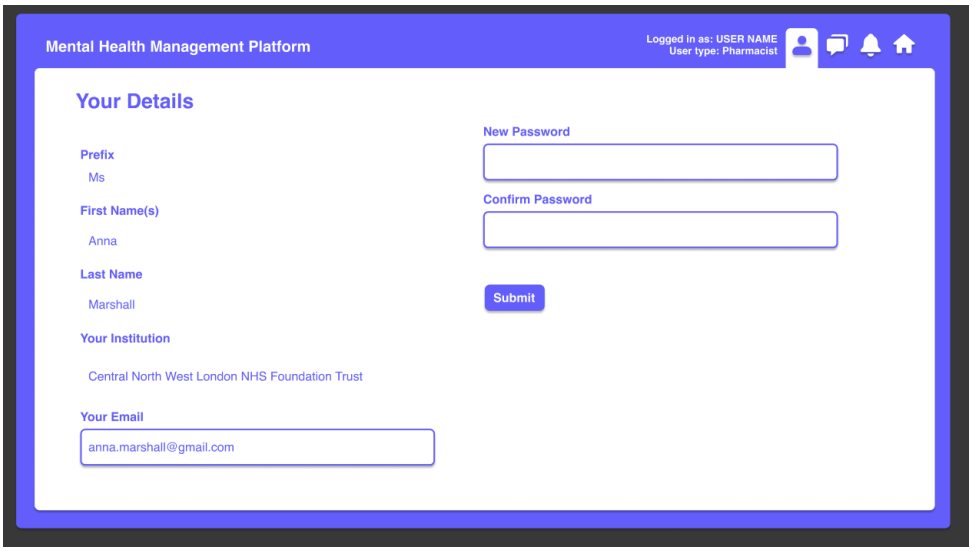
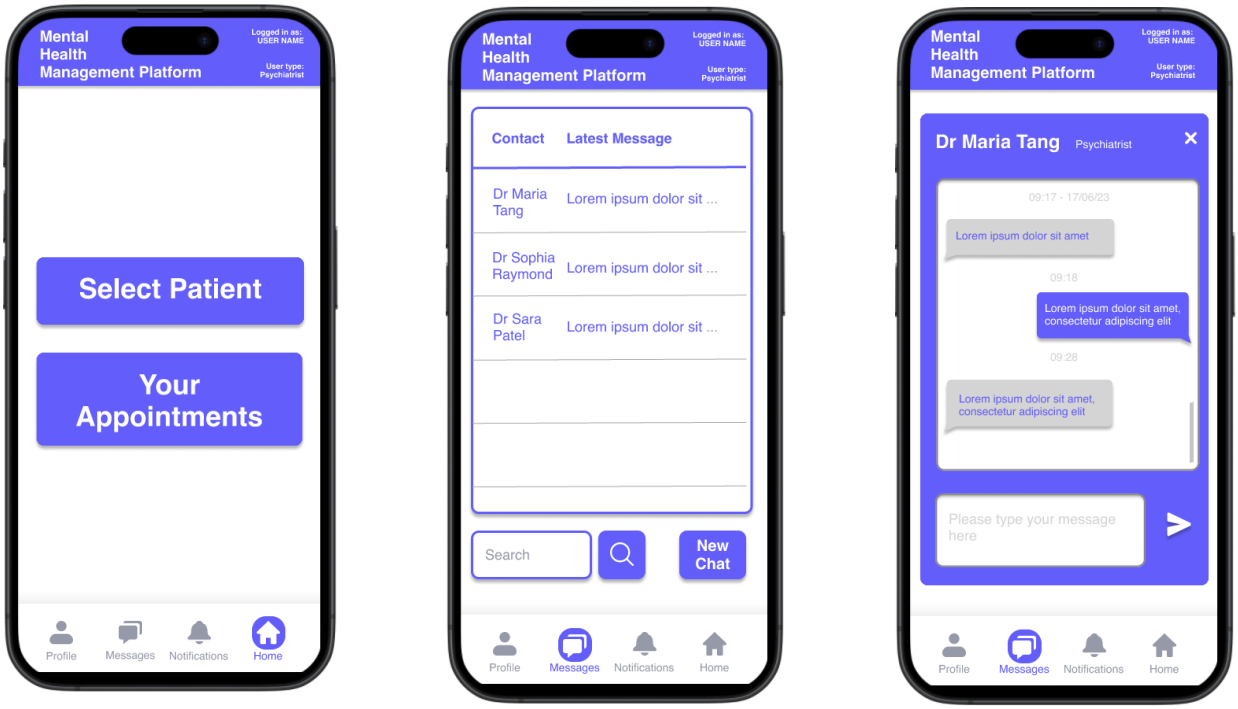


Figure 8.3.0.3: Edit Details - Non-patient View



(a) Main Menu - Mobile **(b)** Messages - Mobile **(c)** Chat - Mobile

Figure 8.3.0.4: Mobile screens side by side

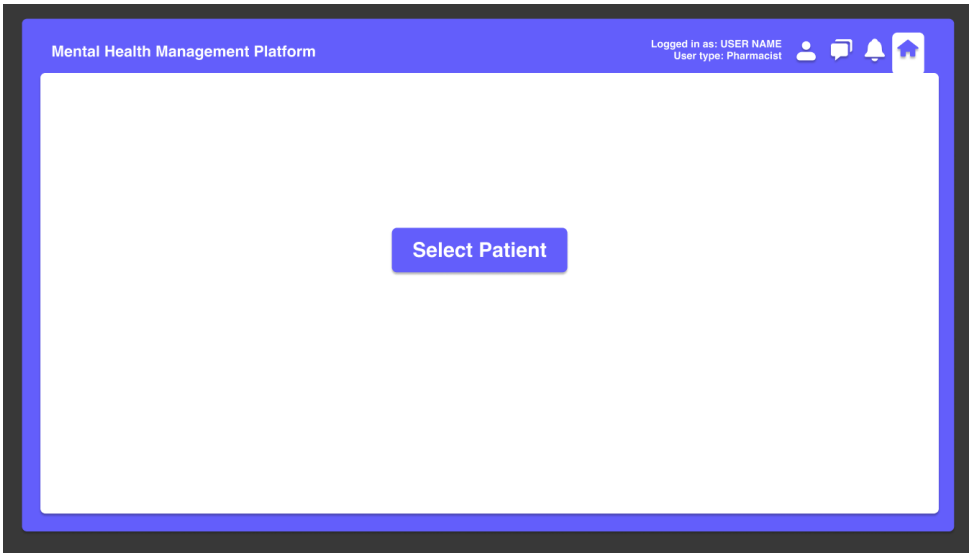


Figure 8.3.0.5: Pharmacist / Nurse Main Menu

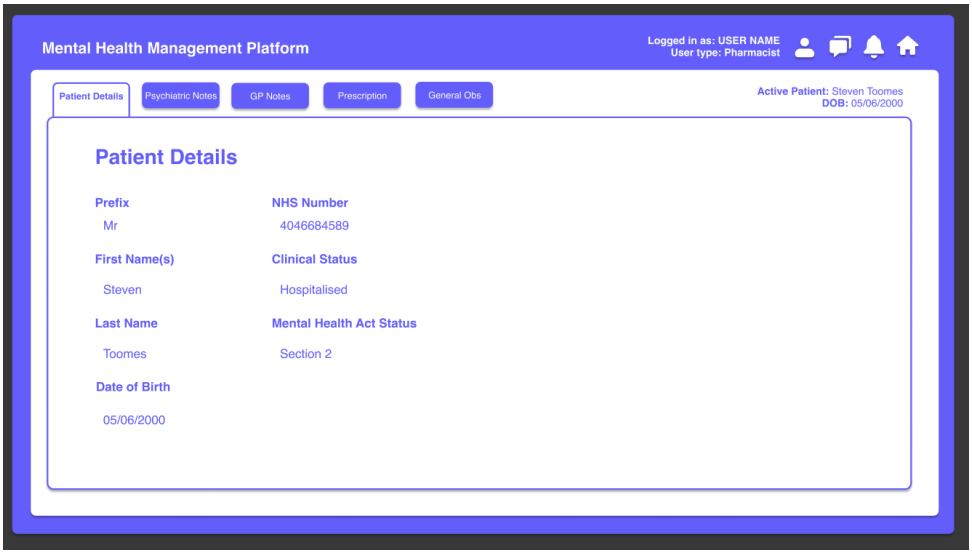


Figure 8.3.0.6: Pharmacist / Nurse Patient Details View

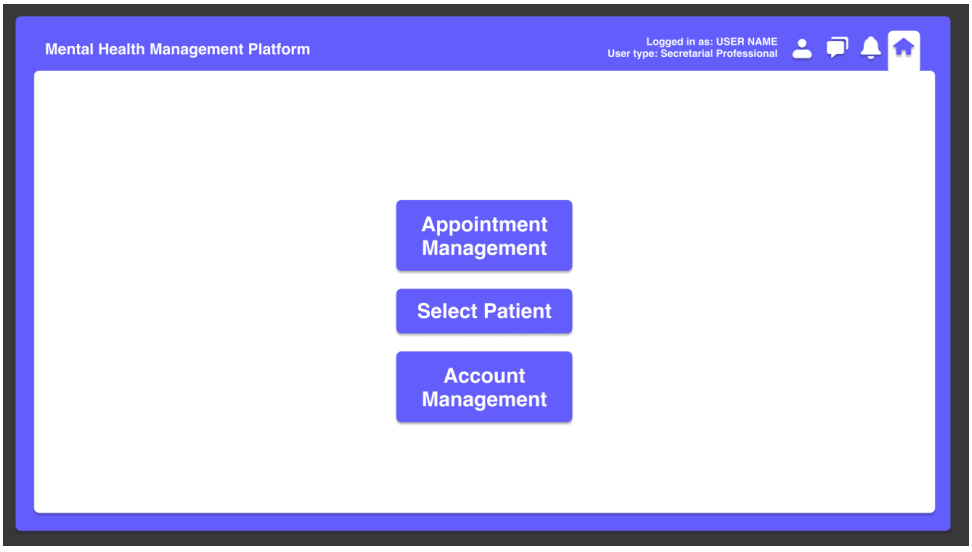


Figure 8.3.0.7: Secretarial Professional Main UI View

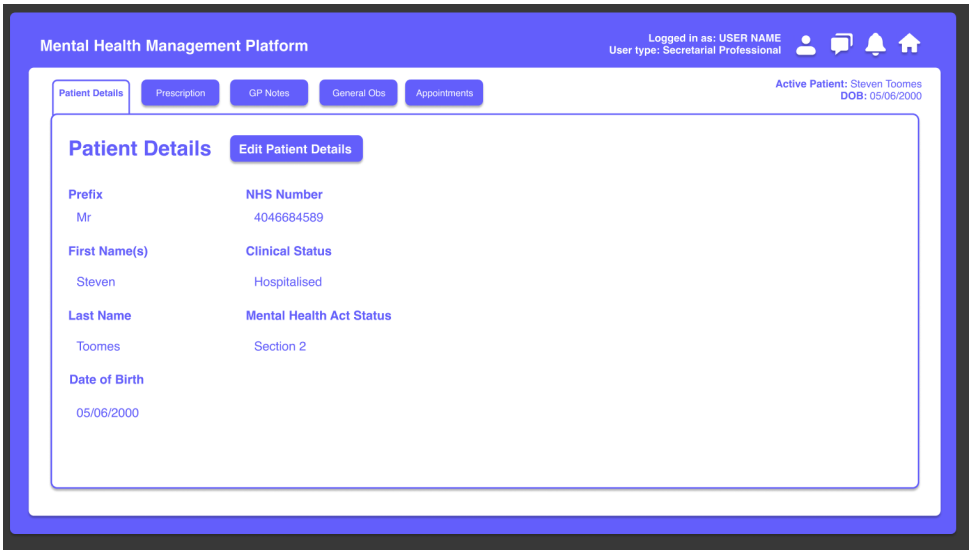


Figure 8.3.0.8: Secretarial Professional View and Edit Patient Details

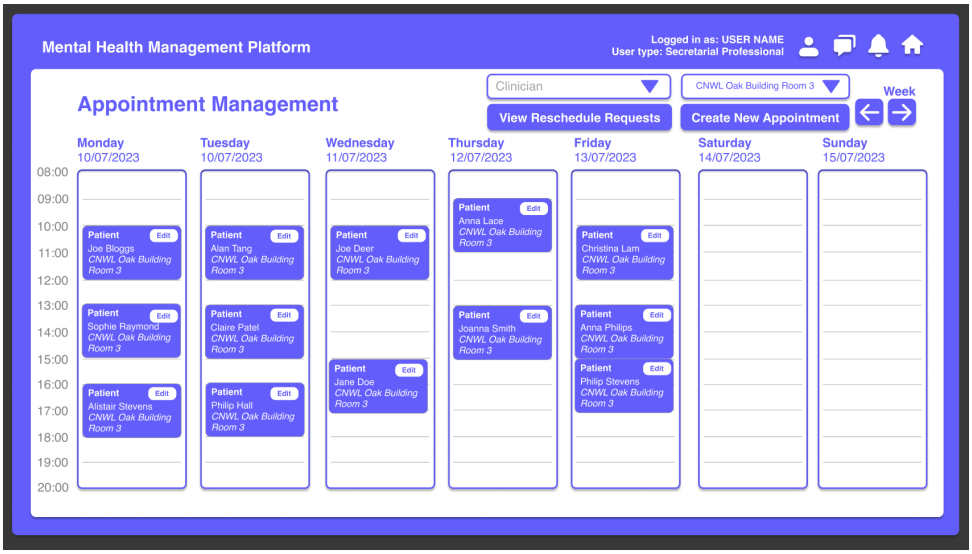


Figure 8.3.0.9: Secretarial Professional Appointment Management (with filtering ability for clinicians and location)

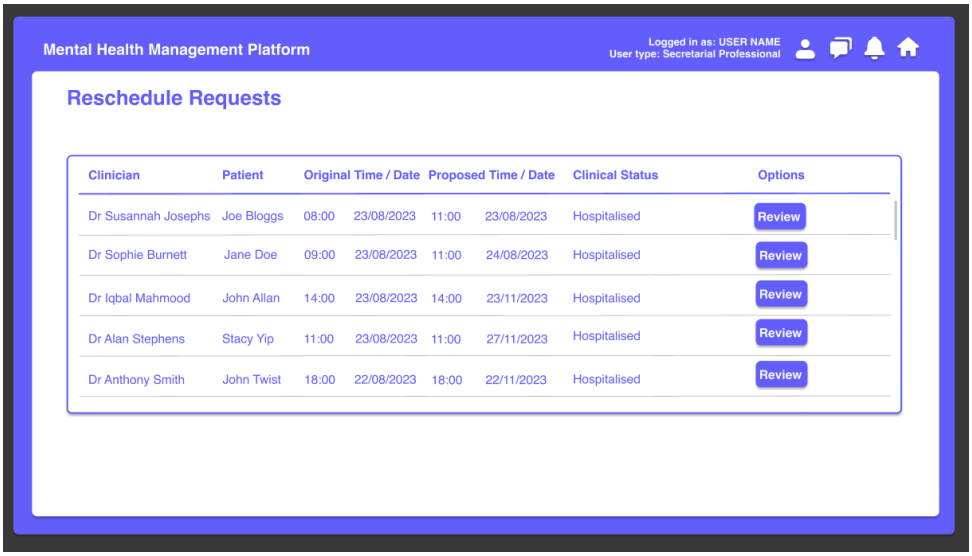


Figure 8.3.0.10: Secretarial Professional Reschedule Requests View

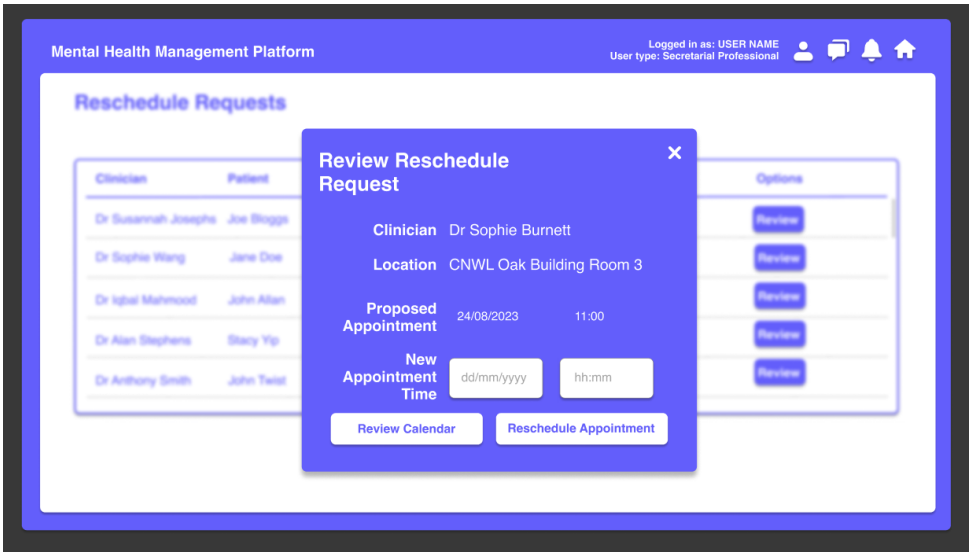


Figure 8.3.0.11: Secretarial Professional Reschedule Request View

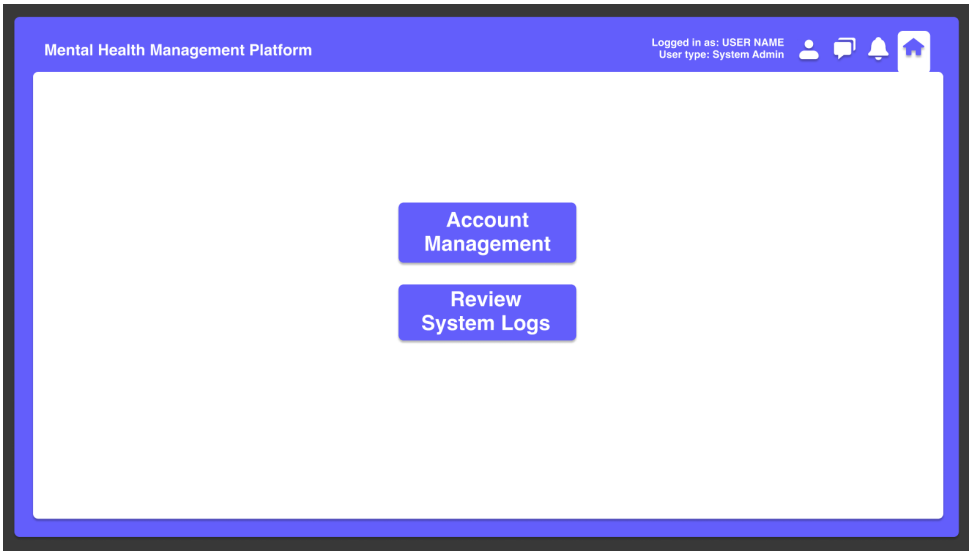


Figure 8.3.0.12: System Admin Main Menu

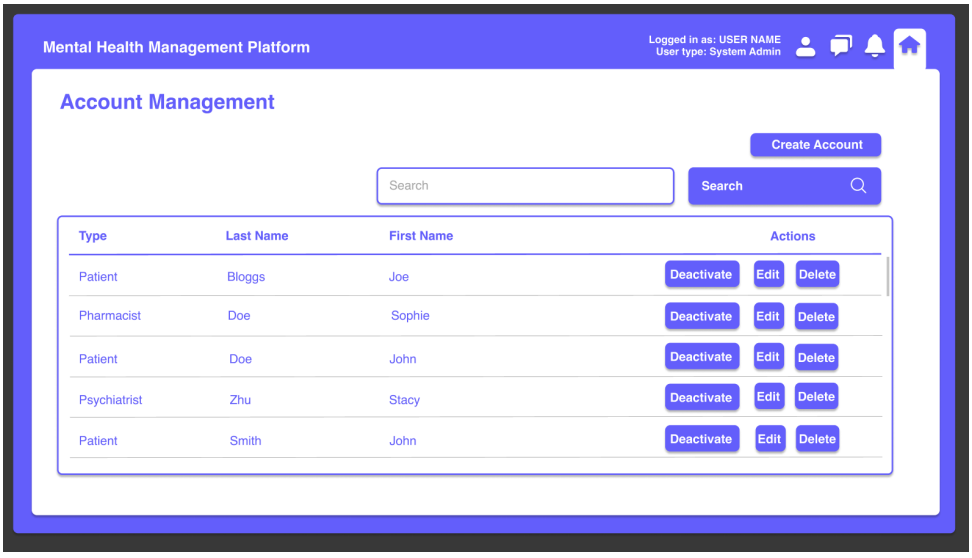


Figure 8.3.0.13: System Admin Account Management - similar to the Secretarial Professional Account Management page

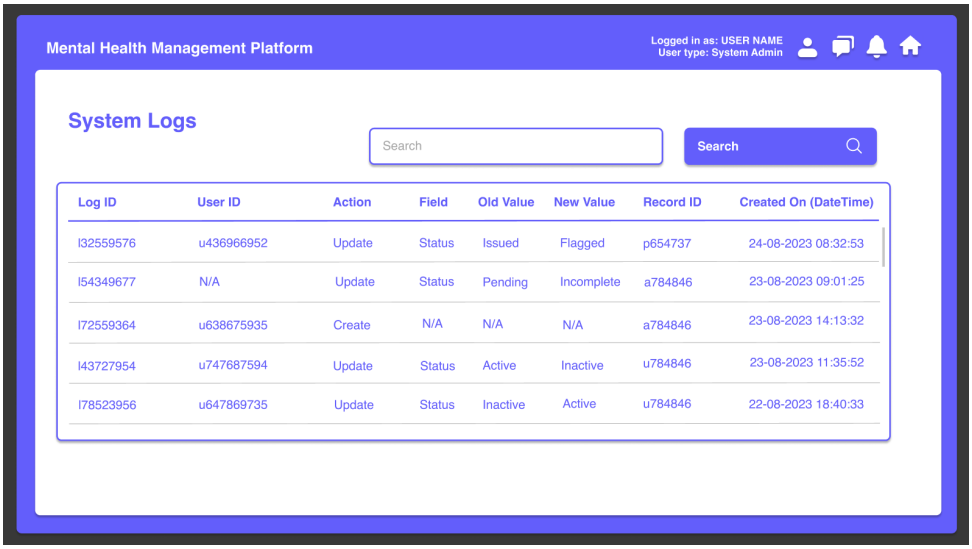


Figure 8.3.0.14: System Admin Logs Review

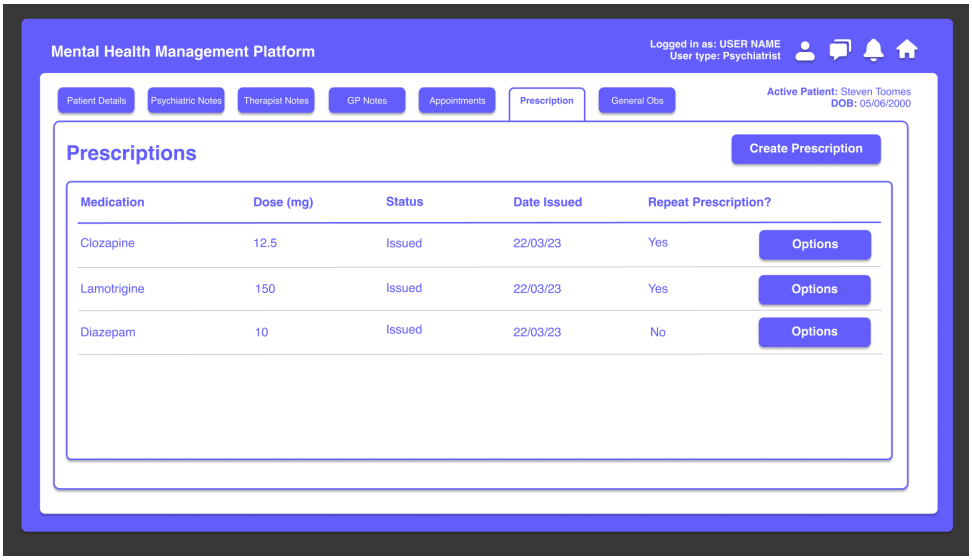


Figure 8.3.0.15: Psychiatrist - Prescription View

Figure 8.3.0.16: Psychiatrist - Patient Details View

References

[1] Karen Epper Hoffman. The pandemic saw the rise of remote mental health care. it may be here to stay, 2021.

[2] British Medical Association. Child and adolescent mental health services: Indexed number of fte doctors in comparison to the number of people in contact with these services. BMA Analysis of NHS Digital Workforce Statistics, NHS Digital Mental Health Services Monthly Statistics, 2023. [Online: accessed 25-March-2024].

[3] NHS England. Mental health services in england. NHS Mental Health Dashboard, 2023. Accessed: 25-March-2024.

[4] Wikipedia contributors. Lorenzo (electronic health record). 2024. [Online: accessed 25-March-2024].

[5] EHR Guide. Epic ehr software: Transforming healthcare it, 2023. [Online: accessed 25-March-2024].

[6] NHS Confederation. Maximising the potential of digital in mental health. NHS Confederation, 2023. Accessed: 25-March-2024.

[7] NHS. Nhs app, 2024. [Online: accessed on 03-April-2024].

[8] NHS. Nhs api catalogue, 2024. [Online: accessed on 03-April-2024].

[9] Ralph Rowland Young. *The requirements engineering handbook*. Artech House, 2004.

[10] Ian Sommerville. *Software Engineering, 10th Edition*, chapter 4. Pearson, 2015.

[11] Chapter 10: Moscow prioritisation, 2014. [Online: accessed on 25-March-2024].

[12] Doug Rosenberg and Kendall Scott. *Use case driven object modeling with UML*. Springer, 1999.

[13] Alistair Cockburn and Lord Cockburn. *Writing effective use cases*. Pearson Education India, 2008.

[14] Russ Miles and Kim Hamilton. *Learning UML 2.0: A Pragmatic Introduction to UML*. O'Reilly Media, Inc., 2006.

[15] Ian Sommerville. *Software Engineering, 10th Edition*, chapter 7. Pearson, 2015.

[16] Mike O'Docherty. *Object Oriented Analysis and Design: Understanding System Development with UML 2.0*. Wiley, Chichester, England ; Hoboken, NJ, 2005.

[17] Robert C. Martin. Principles of object-oriented design, 2024. [Online: accessed 03-April-2024].

[18] Graham Roberts. Snackbar uml class diagram, 2006. [Online; accessed 25-March-2024].

[19] Russ Miles and Kim Hamilton. *Learning UML 2.0*. O'Reilly Media, Inc., Sebastopol, CA, 2006. See Figure 7-20 for use of ”create” and ”destroy”.

[20] Russ Miles and Kim Hamilton. *Learning UML 2.0*. O'Reilly Media, Inc., Sebastopol, CA, 2006. See Figure 7-21 for use of ”ref”.

[21] Object-oriented analysis and design: Understanding system development with uml. Online, 2020. Accessed: 2024-03-29, pp. 498.

[22] Russ Miles and Kim Hamilton. *Learning UML 2.0*. O'Reilly Media, Inc., Sebastopol, CA, 2006. See Figure 7-8 for use of arrow notation.

[23] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional, 3 edition, 2004.

[24] Russ Miles and Kim Hamilton. *Learning UML 2.0*. O'Reilly Media, Inc., Sebastopol, CA, 2006. Refer to Section 8 for use of black-box style.

[25] Jakob Nielsen. 10 usability heuristics for user interface design, 2024. [Online; accessed 25-March-2024].

[26] Yvonne Rogers, Helen Sharp, and Jennifer Preece. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley Sons PT, 6 edition, 2023.