CO3201 COMPUTER SCIENCE PROJECT DISSERTATION

CHRONICCARE: A WEB-BASED APPLICATION FOR MANAGING CHRONIC PATIENTS HEALTH

BY CALEB OSAMEDE EDOSOMWAN

MAY 2025

SCHOOL OF COMPUTING AND MATHEMATICAL SCIENCES,
UNIVERSITY OF LEICESTER

Table Of Contents

Declaration

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s).

Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s).

I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Caleb Osamede Edosomwan

Date: 4th April 2025

Signed: 

Abstract

As of present, healthcare systems have historically struggled to properly manage patients with chronic health conditions, normally leading to fragmentation in the provided healthcare, and generally worse health outcomes for the patients [1].

For this project, I have developed a web-based health application for both doctors and patients, where doctors can efficiently manage their chronic patients, and bridge the consistent communication gap between patients and doctors.

I have developed this system using the Spring Boot MVC framework in the Java programming language with additional tools, for easier third-party access to helpful features and reduced complexity [2].

The project will also highlight problems and issues involved in creating an application that handles highly sensitive information, emphasizing the importance of privacy and security implementation [3].

The successful development of such system will aim to hugely improve prescription and medication management, ease the appointment booking process, and elevate the current methods in which doctors provide consistent and engaging care to their chronic patients, therefore improving the world of digital healthcare as a whole.

1. Introduction

1.1 Background

In the current day, chronic diseases are on the rise. Globally, the number of people living with diseases such as cancer, diabetes, kidney disease, etc, are rapidly rising each year. A study and analysis led by the Health Foundation's Real Centre, predicted that by 2040, 1 in 5 adults will be living with a major chronic disease [4].

Current systems for chronic patient management lack the fundamental features a chronic patient would need, because most general healthcare systems are not tailored to give consistent and engaging care properly [5]. There are technical limitations and gaps in these systems, with issues such as limited health data visibility for patients, and communication inefficiencies [6]. This project is a good opportunity to create a system that fixes these problems and enhances the world of healthcare.

1.2 Aims of the Project

The **main aim** of this project in general is to create a well-functioning web based application to manage chronic patients' health, allowing doctors to provide consistent and engaging care to their patients, with patients also being able to view their health metrics and be actively involved in their healthcare decision-making processes.

The **secondary aim** is to understand and learn how to deal with and handle highly sensitive data in application development. The system focuses on a health issue, so privacy and security of the information must be well-protected.

1.3 Objectives of the Project

The main objectives of this project are as follows:

- Gather information from background research, resulting in relevant insight into what is needed for a well-functioning web application that would effectively manage chronic patients' health.
- Design an interactive web based application for managing chronic patient's health using the Spring Boot MVC framework in Java, creating distinct functional interfaces for both a patient and doctor.
- Implement role-based access control for the system users, ensuring data protection, validation and security.
- Create a functional database system using MySQL to store the relevant data and information for each user **securely**.

- Develop key system features being appointment scheduling, prescription management, and secure messaging to enhance the chronic care management process.
- Implement and undergo various tests for the system, to further learn and develop scientific communication skills, and build on the functionality of the application.
- Learn to account for the issues of privacy and security in developing information systems.

2. Background Research and Literature Review

2.1 History of Digital Healthcare Solutions

The world of digital healthcare solutions became a concept in the mid-1960s, and started to take shape from the introduction of electronic medical records [7]. This laid the foundation of the further use of technology within the field of healthcare and medicine. In the 1980s, researchers and doctors started to believe that computers would be much faster for access to procedure results, and would help caregivers by providing reminders and alerts [8].

Figure 1 – The very first electronic medical record (PROMIS) being used in the year 1977

In the 1990s, a framework known as the Chronic Care Model (CCM) was implemented and aimed to improve chronic disease management. It focuses on patient-centred care, data-driven decision support and the provision of detailed information [9]. Studies proved that this framework led to improvements in treatment outcomes, and it marked a change in how chronic disease management systems are designed [10].

More significant breakthrough for research on chronic healthcare system management came with an analysis conducted by 4 Australian students in 2021. A systematic review on smartphone applications for chronic disease monitoring and management, found that application-based interventions had notable impact on health habits such as

eating and exercise [11]. This proves that application based healthcare systems are highly beneficial to health monitoring.

2.2 Proven Beneficial Features

According to methodical reviews, modern-day digital health care solutions have been proven to have vital functionalities that will improve chronic disease care outcomes. These systems have proven to be particularly crucial, as they utilise tools and functionalities that enhance patient self-management, real-time health monitoring and provide a platform for clear and consistent communication between the patient and doctors [11].

A structured review of digital medicine inventions showed that features such as remote monitoring, digital check-ups or teleconsultations, and personal health dashboards with relevant information, can lead to improved health markers and overall better chronic disease management [12].

Further studies also emphasized that continuous care and secure communication is crucial, as this was highlighted in the COVID-19 era. The sudden shift to online services showed the importance of efficient remote technology in healthcare. It also showed the need for adaptive treatment plans and care for chronic patients, such as remote interactions and flexible management [13]. Learning from this, my system will need to implement effective communication measures.

More research also showed that user engagement contributes to an effective digital healthcare system. Personalised features such as health dashboards to track individual health records and statistics, will improve user engagement [12]. Gamification elements can also support in maintaining user interaction, however I have chosen not to include these as to avoid potential distractions from the main focus of chronic health management. Generally, the studies showed that it is important to also make the system interactive and adaptive, as well as functional.

2.3 Current Limitations

Despite current day chronic disease management platforms being somewhat suitable and defined towards chronic patients, they have significant limitations which in return cause struggles in proper management.

The first notable common issue is the regularity of poor user interfaces. Studies have shown that most of these platforms have user interfaces that are deemed too difficult to navigate, which decreases user engagement [14]. This then causes patients to be less involved in decision-making processes, and eventual decreased use of the system, which leads to worse health outcomes for the patients.

Most platforms are also tailored to handle and manage one chronic disease, instead of being able to manage patients suffering from multiple chronic conditions [15]. Systems that are able to manage multiple chronic diseases and take into account the complexity of them, have proven to lead to better health outcomes for patients.

Current platforms also fail to properly exist and work with electronic medical records (EMRs), causing inefficiencies in data-driven decision making [16]. Either a system that makes use of an in-built medical record or one that can properly integrate EMRs, would lead to efficient health data sharing and collaboration, and in turn make informed decisions to produce better health outcomes [17].

2.4 Privacy and Security for Healthcare Systems

Due to the rise and advancements of technology in healthcare and electronic medical records, robust laws and frameworks have been introduced to implement administrative, physical, and technical barriers to secure patient data [18]. The most significant one is **the HIPAA security rule**.

The HIPAA security rule is a large set of security standards that businesses must follow to ensure that electronic Protected Health Information (ePHI) is collected, stored, and transmitted safely [19]. This rule is holds high importance for many reasons.

Firstly, this rule not only safeguards patients' health data from unauthorized individuals, but also allows healthcare organizations to mitigate the risk of health data breaches. More relevant towards the project, the rule also enables developers to create innovative technologies that improve the quality and efficiency of patient care, while also protecting the data of the patient [20].

To adhere to this rule in the development of a chronic disease management system, various forms of technical safeguards must be implemented. An important one is role-based access control. Strict access control for sensitive data such as the protected health data, would need to be implemented to limit who has access to certain roles that then have access to sensitive patient information [21]. Another way of ensuring data integrity is encryption, as it helps ensure confidentiality during transmission and storage of sensitive patient data in databases [22].

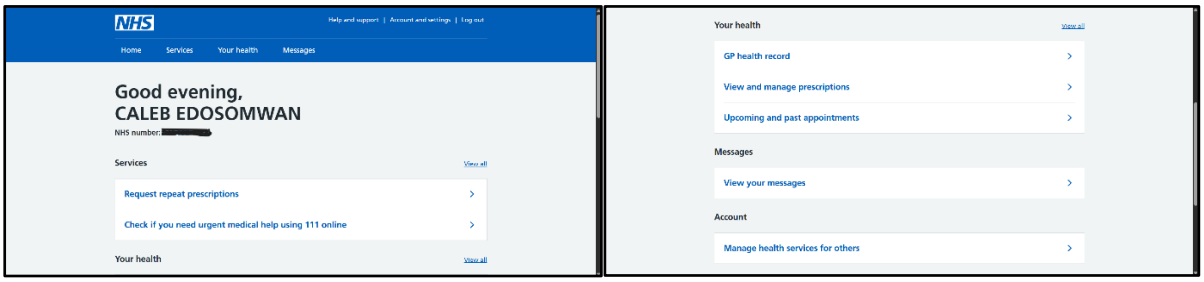
2.5 Current Existing Solutions

Analysing current existing solutions for chronic health care management and similar platforms would be imperative towards my project, as it would give me knowledge of useful features I could potentially implement into my system. This process will also allow me to spot the gaps and limitations in current platforms, enabling me to incorporate the vital missing features into my own.

NHS App

The NHS app is one of the most widely used application systems in the UK for managing health information, ordering prescriptions, and appointment management [23]. This makes it suitable for analysis and comparison to a theoretical system for patient management.

Dashboard and Messaging System Features



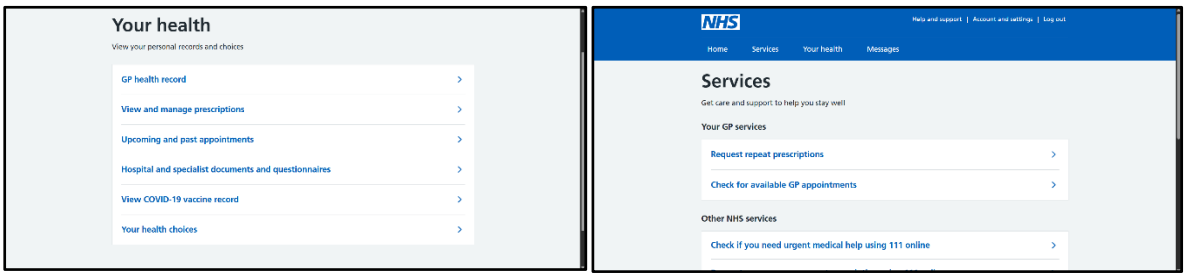
Figures 2 and 3 – Dashboard of a logged in patient on the web version of the NHS application

Analysing the dashboard at first glance for a patient, it has multiple useful features for self-patient management. You can request repeat prescriptions, view and manage prescriptions, view upcoming and past appointments, and manage messages.

However, the messaging system does not actually allow you to directly message your GP or surgery, as it only permits you to view messages from the surgery. This creates an opportunity for fragmented communication, as patients cannot easily and directly contact their doctor through the system.

Another crucial issue with the dashboard is that it is neither data-driven, nor patient-centred. There are no features to show the patient any details or trends on their current health conditions. The dashboard is also not very interactive, with a dull and poor user interface. This leads to less interaction from patients, less involvement in decision-making, and eventual worse health outcomes.

Health and Services Features



Figures 4 and 5 – "Your health" and "Services" tabs of the NHS application

Analysing the health and services features, the health tab allows you to view a comprehensive summary of your health record, with allergies, medicines, past test results and documents. This is a very useful feature as it allows patients to be actively aware and involved in the progression of their health and healthcare [24].

The services tab also gives patients information on emergency services they can directly contact if they are in need of urgent medical care, which is very important because it allows patients to respond quickly in emergencies.

Epic MyChart

Epic MyChart is an electronic health record system used by 150 million patients across the US, and is used by most of the top hospitals and medical schools [25].

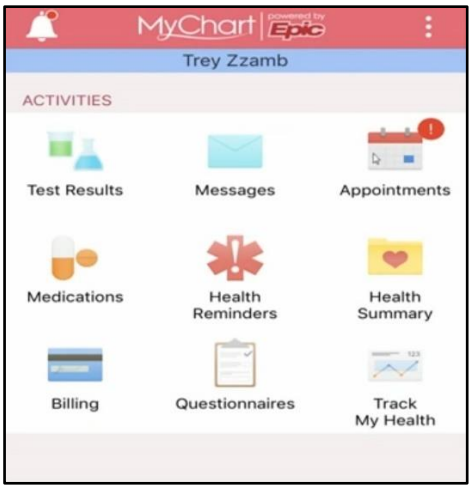


Figure 6 – Epic MyChart mobile dashboard

The system allows for various useful functionalities, as patients can view medical records, test results, manage appointments, get health reminders, track their health metrics and communicate securely with their doctors [26]. The system also has a user-friendly interface, and encourages active participation with its interactive features.

However, the system has a few limitations. One of these is that the system is not properly catered to users who may suffer from disabilities and have difficulty using screen readers, causing accessibility issues [27]. This is crucial in a healthcare system, as proper accessibility features will ensure that all patients can access healthcare services without facing any unnecessary issues [28].

Summarising the background research and current system analysis vaguely, I discovered the gaps and limitations from the existing chronic patient management platforms and deduced that it is important that my system needs to prioritise real-time updates, personalised care, and consistent engagement and communication, while also prioritising patient data confidentiality and secure data exchange.

3. Requirements Analysis and Specification

3.1 Functional Requirements of the System

Following the extensive research I conducted, I was able to create a list of the main functional requirements my system will need to implement for the users, in order to make an efficient and engaging chronic health management system.

During development, I modified the original list of functional requirements for my system from my interim report, by adding more features I deemed were necessary. These changes were added following feedback I had received during my interview, and during user feedback sessions.

3.1.1 User Management

- Users will be able to register and login to the system securely, with appropriate authentication and verification methods.

Role-based access control ensures that users have access to functionalities based on their roles. For example, doctors can view and access all their patients' health records, whereas a patient can only view their own. This streamlines functionality and ensures data security, while reducing the risk of unauthorised access [29].

3.1.2 Patient and Doctor Dashboards

- Patients will be able to view a home screen dashboard with personalised tools and information regarding their healthcare.
- Patients will be able to monitor key health metrics such as blood levels and other vital signs of health.
- Patients will be able to view visual tools to offer a trend analysis over their chronic health status.

- Patients will be able to view a prescription guide, giving them precise and detailed information and instructions about how to take their assigned prescriptions.
- Patients will be able to create custom medication reminders for their assigned prescriptions.
- Doctors will be able to view a home screen dashboard with personalised tools and information regarding their patients.
- Doctors will be able to view their patients' medical records and also monitor key health metrics, just like their patients.
- Doctors will be able to assign themselves patients to manage.
- Doctors will be able to view recent updates regarding patients, and see if any patients require immediate attention.

An interactive dashboard with tools and information tailored to each patient, will allow them to understand their health path and trajectories, and efficiently collaborate with doctors to make data-driven decisions. It will also ensure patients are actively involved and engaged in their own chronic health care management, rapidly improving patient outcomes [30].

3.1.3 Appointment Management

- Patients will be able to book appointments with doctors.
- Patients will be able to reschedule appointments with doctors.
- Patients will be able to cancel appointments with doctors.
- Patients will be able to receive automated reminders of appointments.
- Doctors will be able to view and accept appointments with patients.
- Doctors will be able to view a calendar, showing a detailed schedule of appointments they have for each day.
- Users will be able to view a tab of pending appointment requests, upcoming appointments and completed appointments.

Regular checkups are an essential for chronic health monitoring, as they ensure any health complications are identified from early, and ensure treatment is effective [31]. Therefore, a feature that easily allows efficient appointment scheduling will be a necessity.

3.1.4 Communication System

- Users will be able to access a secure messaging system between both parties exclusively, to allow them share updates, ask questions, and give or receive any necessary information.

- Users will be notified about updates, including appointments, prescription refills and messages.

An efficient communication system will be essential, as it will ensure patients remain informed about their health. Healthcare for chronic patients often requires feedback loops and regular updates [32], so this will avoid the consistent communication barrier between in-person appointments, and will enhance patient-doctor relationships.

3.1.5 Prescription Management

- Patients will be able to view a comprehensive list of their current assigned prescriptions.
- Patients will be able to request refills for their current prescription directly from the system.
- Doctors will be able to create prescriptions for their assigned patients.
- Doctors will be able to review and approve/deny prescription requests from patients.
- Users will be able to access a prescription record and track their medication history.

Having an easily accessible record of all medications will also make way for informed decisions relating to care adjustments [33]. This feature will ease the process of medication management, and will ensure that patients have access to necessary medications without any delays.

In summary, these functional requirements will provide a comprehensive and secure health system that maximises patient care and streamlines healthcare delivery. A system based on these requirements will give chronic patients the resources they need to properly manage their conditions, and give doctors the resources they need to efficiently manage their patients.

3.2 Non-Functional Requirements of the System

My system will aim to follow multiple non-functional requirements, which are crucial to ensuring the system meets the project objectives. They make sure the system is produced at the highest quality, and guarantee the overall success of the project [34].

Security requirements

- My system will implement user authentication and role-based access control to protect patient and doctor tools and resources
- My system will protect and secure personal data through encryption methods

- My system will implement a session timeout of 30 minutes to prevent unauthorised access from abandoned sessions

These requirements will ensure that my system is fully secure. Compromising these security requirements will pose significant risks towards exposing sensitive patient data, allowing unauthorized access, and regulation breaches [35].

Performance requirements

- My system will load pages for no more than 3 seconds in normal working conditions
- My system will execute database queries in no more than 400 milliseconds in normal working conditions
- My system will support the use of multiple concurrent users with no drop in performance

My system will need to perform at a high quality, which is shown to improve user satisfaction, workflow efficiency, and the general effectiveness of the system [36].

Usability requirements

- My system will have fully a functional, responsive and interactive user interface, compatible with multiple desktop browsers
- My system will allow for easy navigation through the main features and pages of the interface
- My system will make use of efficient error and exception handling

Usability and a fluid user-experience is essential for a health system, as this will enhance overall patient engagement, and produce better outcomes [37].

Scalability requirements

- My system's architecture design will support horizontal scaling to handle increasing workload and user base
- My system's database design will allow for efficient and fast patient and appointment data retrieval as the number of records increase

These requirements will improve the scalability of my system, which will be crucial for handling increasing workload and user demands, without degrading the performance of the system [38].

These non-functional requirements will allow me to meet the objectives of the project, while also ensuring that the system of a high quality, usable, and a long-term success.

4. Design of the System

4.1 Architecture Overview of the System

4.1.1 Spring Boot MVC framework

My web based chronic health management system will make use of the Spring Boot framework, and will use the Model-View-Controller (MVC) pattern. This approach will allow me to separate my system into components that work together to handle the system logic, data management and user-interface design.

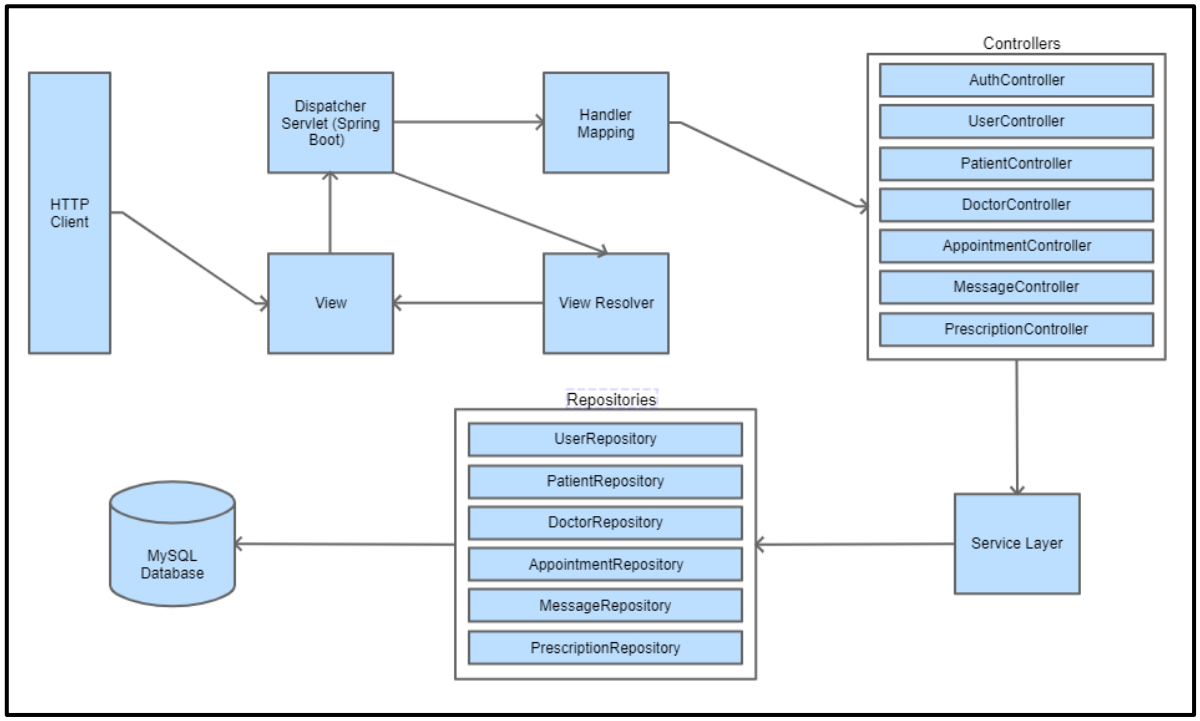


Figure 7 – The MVC system architecture showing the efficient backend workflow of the system

The first entry point for the user will be through the HTTP client, as the users are greeted to the first interface through the web browser. The view then allows users to interact with the system via front-end webpages, handled by HTML templates. The diagram then shows the normal MVC pattern as requests from the client are handled by the dispatcher servlet, which then routes the request appropriately. The servlet then uses handler mapping to determine which controller is meant to process the request. The controller will then call the service needed for the right repository. The repository will then be responsible for database interaction and perform the appropriate CRUD operations.

4.1.2 Thymeleaf

Thymeleaf is a modern Java template engine, that allows for easy management of web pages in Spring boot, and is helpful for user-interface design in independent work [39].

My system will make use of Thymeleaf by using its design-time placeholders to display dynamic content for forms and display boxes on each webpage, mainly the dashboards. It will also use its efficient HTML page management system. This will create an efficient bridge between my backend code logic and my frontend HTML presentation.

4.1.3 MySQL

MySQL allows for easy integration, simplified data access, and support for relational databases. This is important for applications like mine that require strong data consistency [40].

The SQL database in my system will be responsible for storing all critical data. This will include user details, health data and metrics, appointment schedules, messages and prescription data.

Each repository will communicate with the database through custom queries when requested. This will ensure data is well protected, and will facilitate seamless data storage and easy data retrieval.

4.1.4 Spring Security

Spring Security is a highly customizable authentication and access-control framework, offering significant support for user authentication and authorization [41].

My system will use this for the login system, registration system and role-based access control for a patient and doctor, to limit who has access to certain roles that then have access to sensitive patient information. The security design framework will be explained further in the 4.5 Security Design section.

4.2 Component Design

This section will break down in detail how all the interconnected components in my system will work together in the Spring boot framework.

4.2.1 Models

The model in my system will handle the application's data logic, responsible for storing necessary data, and will define the structure of all the interrelated relationships of data in the system.

My system will consist of various model classes, that will hold the relevant information needed for each part of the system. The model classes in my application will include:

- **User:** This class will handle user authentication. It will hold core user information, being the username, password and role.
- **Patient:** This class extends from the User class, contains patient-specific fields. and will hold relevant patient information. It has a many-to-one relationship with the Doctor class because multiple patients can be assigned to a doctor. It also has a one-to-many relationship with the Prescription class because a patient can have multiple prescriptions.
- **Doctor:** This class extends from the User class, contains doctor-specific fields. and will hold relevant doctor information. It has a one-to-many relationship with the Patient class because a doctor can manage multiple patients. It also has a one-to-many relationship with the Prescription class because a doctor can create and assign multiple prescriptions.
- **Appointment:** This class will handle appointment scheduling and data between doctors and patients. It contains fields for the appointment date, time and status. It has many-to-one relationships with both the Patient and Doctor classes because both users can have multiple appointments.
- **Prescription:** This class will handle prescription management and data, and has many-to-one relationships with the Patient class and the class of the prescribing Doctor.
- **Message:** This class handles the messaging system and all message-related data. The structure allows for flexibility between both user types.

4.2.2 Controllers

The controllers in my system will handle the application's functional logic. They will receive requests from the dispatcher servlet and process any input data from the user.

My system will consist of various controllers, that will each handle a specific aspect of the system. The controllers in my application will include:

- **AuthController:** This controller will handle methods and requests for logging in, registering, authentication and access control by redirecting users to their respective dashboard based on whether they are a patient or doctor.
- **PatientController**: This controller will manage functionalities related to patient accounts, mainly dashboard features such as viewing health metrics and trend analysis.
- **DoctorController**: This controller will manage functionalities related to doctor accounts, mainly dashboard features such assigning patients, viewing recent updates and viewing patient information.
- **AppointmentController:** This controller will handle requests for viewing, booking, rescheduling, cancelling, confirming and completing appointments.

- **MessageController:** This controller will facilitate communication between patients and doctors, processing messages sent by users and passing them through the service layer for further processing or storage. It will handle requests for viewing, composing and sending messages.
- **PrescriptionController**: This controller will handle requests for creating, managing and viewing prescriptions, as well as requesting refills, approving refill requests and viewing prescription history.

The structural design of my controllers changed during the development of my system, by removing the HealthRecordController and adding separate Patient and Doctor controllers. The health record system was eventually handled by a DTO, and the new patient and doctor controllers are to handle the separate and unique dashboard controls of each type of user.

4.2.3 Services

The services in my system will be responsible for acting as a bridge between the controllers and repositories in my system, managing business logic related to the model classes. Each model class will have its own corresponding service class, responsible for retrieving model specific information and interacting with the controllers.

There is an extra service class called **PatientUpdateService**, that is responsible for tracking patient activities and relaying the information to the doctor controller for the Recent Updates feature.

4.2.4 Repositories

The repositories in my system will be responsible for interacting with the MySQL database. They will conduct low-level database operations by providing a layer of CRUD (Create, Read, Update, Delete) functionality. The Spring Data JPA will simplify the interactions through in-built interfaces. Each model class will have its own corresponding repository class, responsible for calling the CRUD operations to work with the database.

4.2.5 DTOs

My system will make use of data transfer objects (DTOs) for the health record feature, patient statistics feature, and to display appointment information. The **AppointmentDTO** will store patient and doctor information, as well as formatted date and time values, for use by the appointment controller. The **PatientStatisticsDTO** will aggregate information needed regarding patient counts, attention requirements, and pending appointments for the patient statistics feature on the doctor dashboard. Lastly,

the **PatientUpdateDTO** will gather recent patient activity data around the system, and relay it back to the doctor service for the recent updates feature. The use of DTOs in my system will improve the application performance, and will make it easier to manage data transfer without complex entity relationships [42].

4.3 Data Model Design

For my system, data integrity, consistency, privacy and security are crucial and must be managed at a high level.

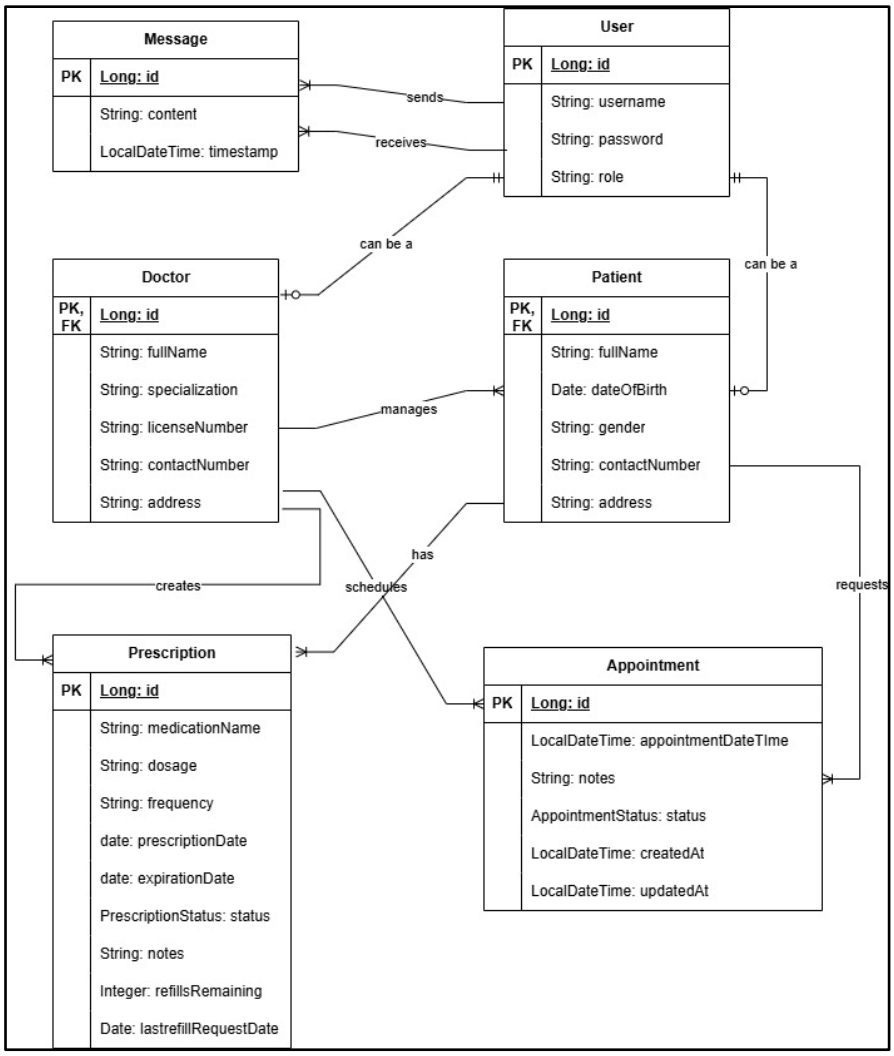


Figure 8 – Entity-relationship diagram of the system's data model

As shown in the entity-relationship diagram above, the relational data model follows a very structured and cohesive design. Both the Patient and User classes are extended from a base User class with a joined inheritance strategy. This allows both types of users to have appropriate unique attributes, while maintaining a good authentication base and ensuring quality query performance [43]. The relationships between the model classes allow doctors to effectively manage patients, and both users to manage

prescriptions and appointments. The relationships between the User and Message classes allow for flexibility in sending and receiving messages for both user types.

This data model structure for my system will guarantee that data is consistent and accurate, while maintaining referential integrity, preventing anomalies, and ensuring that data stays at the highest quality over time [44].

4.4 Frontend Design

The views in my system will represent the user interface and interaction point for the users. They will present data to the users and also take input from users where necessary. The view will function through HTML pages.

In the course of developing my HTML pages, I will make use of Thymeleaf, JavaScript, CSS properties and additional APIs to enhance my pages. These will give my pages interactive layouts, form inputs, data tables, and visual indicators. The views will aim to ensure seamless user interaction and overall enhance the user experience.

I decided to use inline CSS and JavaScript for all my pages for two main reasons. Firstly, this approach provides easy contextual awareness to whoever is reading the CSS script and HTML page [45]. Secondly, it prevents the common issue I have struggled with when importing two different stylesheets into the same page, of the page recognising only one instead of the two. In this scenario, that would be my own custom CSS styles for the design of the page, and the Bootstrap stylesheet for the card and button designs.

The main views in my application and their respective HTML pages will include:

Login and register view

- **login**: This will be the user login page and will contain a form with authentication. Patients and doctors will be required to enter their details, and JavaScript functions will then validate the data.
- **register**: This will be the user registration page and will contain a form with authentication. Users can choose to either register as a patient or doctor, and then fill role-specific fields for registration. JavaScript functions will then validate the data.

Dashboard view

- **patient-dashboard**: This will be the main dashboard interface for the patient, showing a summary of personalised information, accessibility features, health metrics, a prescription guide, medication reminders, and buttons to access the appointment, prescription and messaging features. JavaScript will handle the metrics graph, reminder feature and prescription guide.

- **doctor-dashboard**: This will be the main dashboard interface for the doctor, showing a summary of personalised information, accessibility features, patient statistics, recent updates, the "assign a patient" feature, and buttons to access the appointment, prescription and messaging features.

Appointment view

- **patient-appointments**: This will be the appointment management page for the patient, with a tab to request an appointment for a time and date, a tab to view pending, upcoming and past appointments, buttons to reschedule and cancel appointments, and information on how to book one. JavaScript will handle all functions.
- **doctor-appointments**: This will be the appointment management page for the doctor, with calendar integration for an appointment schedule, a tab to view pending, upcoming and completed appointments, and buttons to accept, decline, cancel and complete appointments. JavaScript will handle all functions.

Messaging view

- **message**: This will be a centralised messaging history page tailored to each user, showing sent and received messages for the user, and a button to compose a message.
- **compose-message**: This will be a form to send a message, allowing users to choose the recipient, and enter the content of the message.

Health Record view

- **patient-details:** This page will display a record and detailed summary of a patient's health data and metrics, prescription history and appointment history.

Prescription view

- **patient-prescriptions:** This page will display a detailed list of all past and current prescriptions for the patient, with a button to request prescription refills.
- **doctor-prescriptions:** This page will display a detailed list of all past and current prescriptions a doctor has issued, with buttons to view prescription refill requests and create prescriptions.
- **doctor-new-prescription:** This page will be a form for doctors to create a new prescription, with fields to select the patient, medication, dosage, expiration date, and refill amount. JavaScript will handle form functions for medication information.
- **doctor-refill-requests:** This page will show a detailed list of the doctor's pending refill requests, with buttons to approve and deny refill requests.

Use Case Diagram

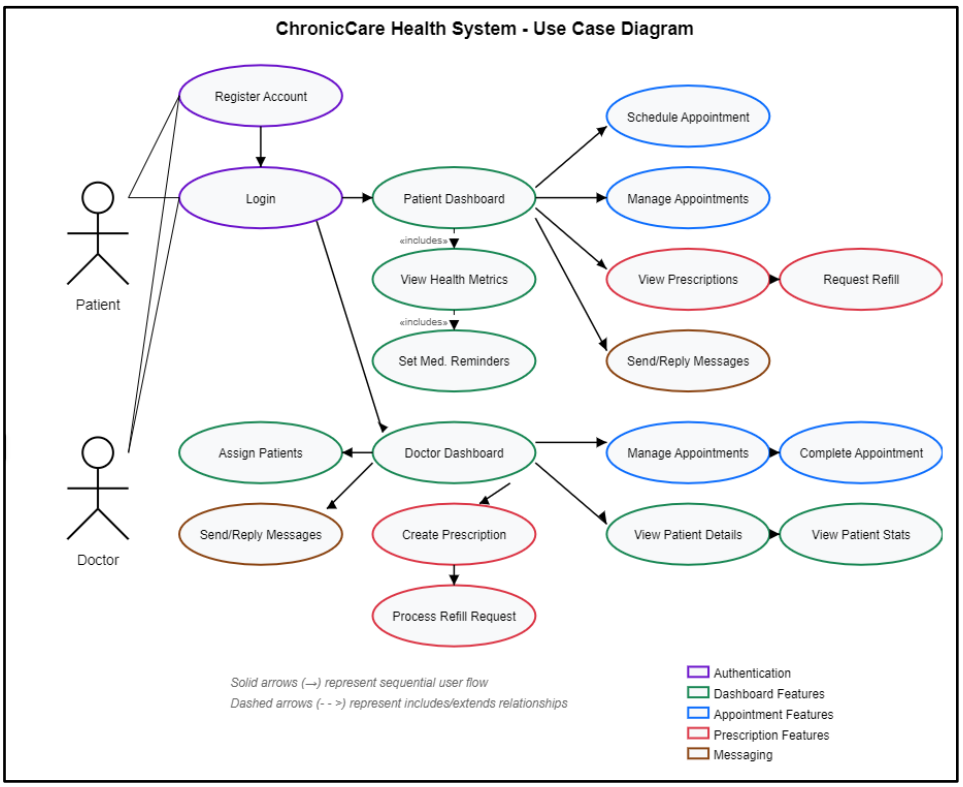


Figure 9 – Use case diagram showing the system's functionality from a user perspective

The use case diagram above shows the sequential flow and progression of functionality as a user navigates through the features of the system, all based on their role.

This structured flow of features mirrors healthcare practices and interactions in the real world, maximising its usability and efficiency [46].

4.5 Security Design

My system will make use a robust and flexible security model framework, based on **Spring Security**. The security model will consist of multiple files and components responsible for the general authentication, authorization, access control, and encryption functionalities of the system.

The files and their functions and responsibilities are:

- **SecurityConfig:** This class will be the main security configuration file, that will make use of Spring Security's signature functions. The class will :
    - Create beans for authentication providers based on the in-built UserDetailsService
    - Define the URL path patterns the system will take once authentication is completed after login

- Create a security filter chain to permit certain requests after a user has been authenticated
- Define role-based access control by identifying the role (Patient or Doctor) a user has after a login request, and ensures the URL begins with /patient or /doctor
- Use the BCrypt password hashing algorithm to encrypt passwords before they are stored in the database

BCrypt is an extremely secure password hashing algorithm, based on the Blowfish cipher. It works by creating a unique "salt", which is a bunch of random data used as an additional input to a one-way function [47]. BCrypt then combines this salt with each password, and hashes the result through loads of iterations. This method produces a fixed-length hash that is extremely difficult to reverse, hard to crack, and protects the data from brute-force attacks [48].

- **CustomUserDetails:** This class will make use of Spring Security's user details class. It will provide the base security methods that Spring Security needs.
- **CustomUserDetailsService:** This service will make use of Spring Security's user details service. It will load a user by username from the database during authentication, and convert my User class entities into Spring Security's user details objects that contain authentication information.

This structured design of the security model will effectively separate the patent and doctor functionalities, while providing a centralised authentication framework, protecting the system against malicious and unauthorized attacks, and ensuring the confidentiality and integrity of the critical and highly sensitive health data.

The design and application structure of my system provides intuitive user interaction points and efficient data handling, ensuring a comprehensive system workflow that supports chronic patient health management for both the doctor and the user, while keeping the system fully secure.

4.5 Alternative Design Approaches

I considered using Spring WebFlux to make the application more efficient and handle more requests, but it was not necessary for the application scale. I also considered designing my system with a RESTful API backend, with a completely separate frontend implementation. This would have improved the scalability of my system, but the implementation would have been a lot more complicated and prone to error. My eventual design approach supports straightforward implementation, and aligns with the security requirements for healthcare data management.

5. Implementation, Development and Walkthrough

5.1 Development Tools

5.1.1 IDE (Integrated Development Environment)

The IDE I chose to use to develop my system in is **Intellij IDEA**. Intellij offers smart code completion and suggestions, and has in-built tools for quick error debugging [49]. These tools are proven to help write code quickly and efficiently, which will bring significant benefit to the quality of my system.

5.1.2 Build Tool

I will use **Gradle** as the build tool during the course of my software development. Although I believe Gradle has some inefficiencies and is normally unpredictable, I am more familiar with its use and debugging, as opposed to other build tools.

5.2 System Implementation and Walkthrough

This section will give a complete and thorough walkthrough, explaining the tools and features of the system from an end user perspective and will show how the structured design of the system was implemented.

Colour theme

Interface	Main Colors
Login and Registration Interface	#2575fc (Blue), #ffffff (White)
Patient Interface	#6a11cb (Purple), #ffffff (White)
Doctor Interface	#198754 (Green), #ffffff (White)
Message Interface	#8B4513 (Brown), #ffffff (White)

Figure 10 – Table showing the system's colour theme for each interface

My system uses a distinctive colour theme to separate the patient and doctor interface. All pages use a colour gradient scheme with white, to enhance the readability and legibility of the text, while keeping a visually appealing interface. The use of brighter colours are also more attention-grabbing, efficiently highlight critical information, and ease user navigation and understanding [50].

5.2.1 Login and Registration Implementation

The first page the user is greeted with is the login page, with appropriate logo branding and a welcome message. The user is then requested to login via a form, with the option to register if they do not have an account.

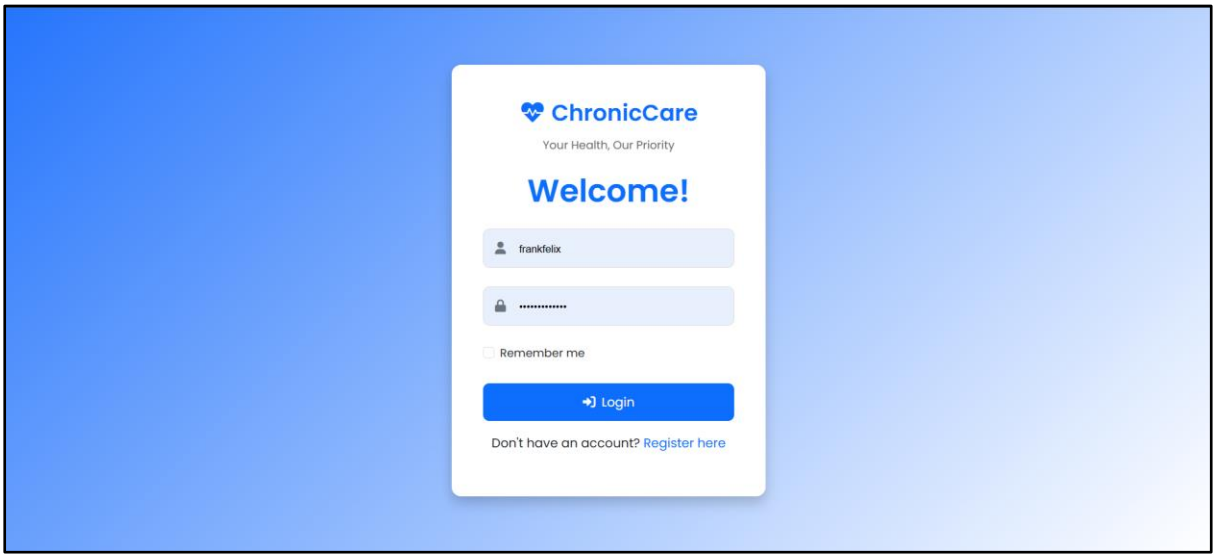


Figure 11 – Login page

The “Register here” link directs the user to the central register page for both patients and doctors.

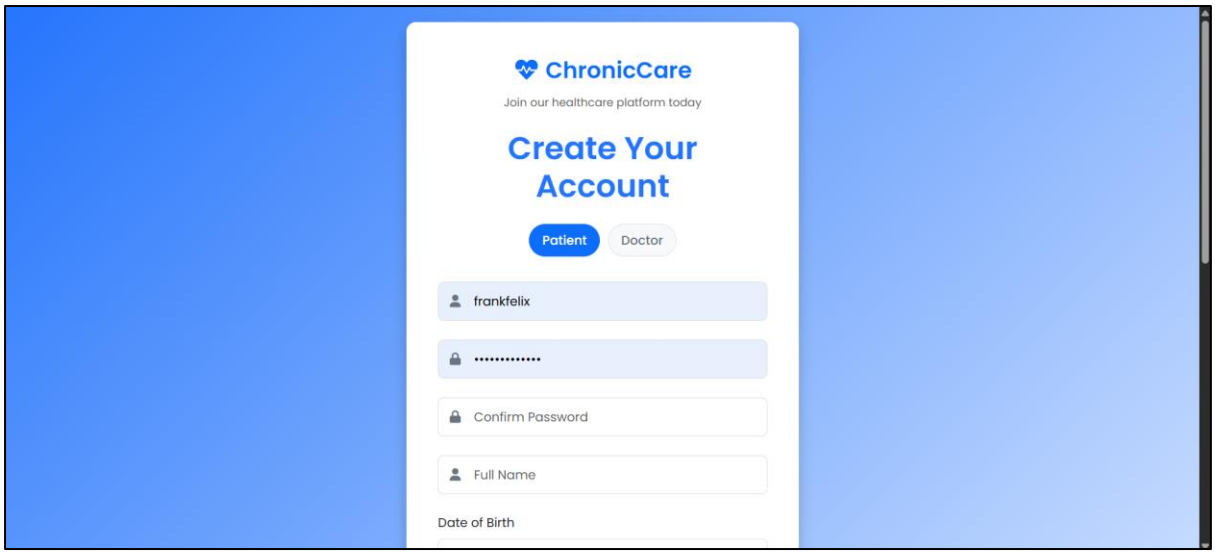
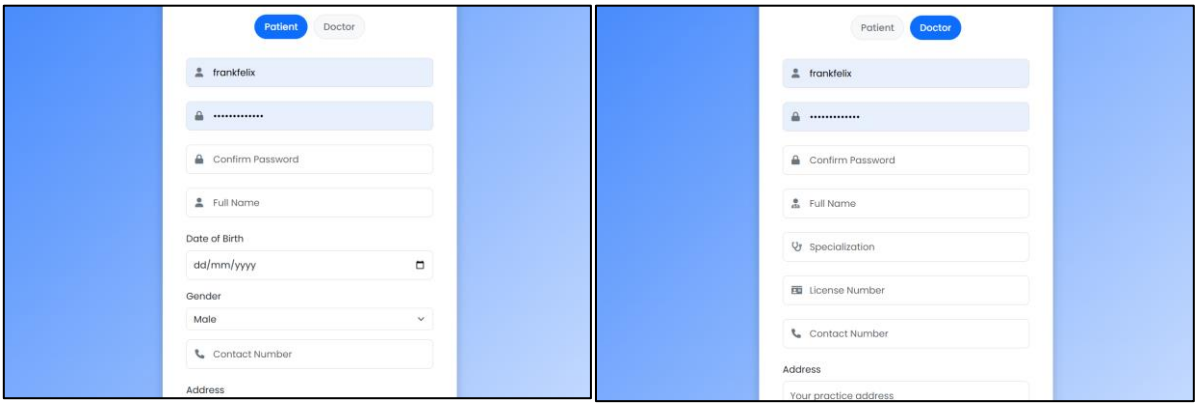


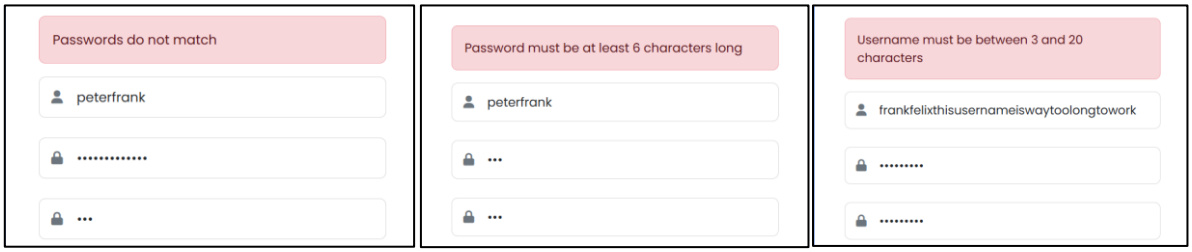
Figure 12 – Register page

Users can toggle a button to log in as either a patient or doctor, with role-specific fields to enter the necessary information for the user.



Figures 13 and 14 – Role toggle functionality

The “Create Account” button is not unavailable until all required fields have been entered with valid information. The username and password fields also have appropriate validation methods, that ensures only valid data is accepted.



Figures 15, 16 and 17 – Validation functionality

The user can then register after validation checks have been passed, along with a link to return to the login page if needed. Upon successful registration, a user is then automatically redirected back to the login page. The user details are then stored in either the Patient or Doctor table, and the login details are stored in the User table.

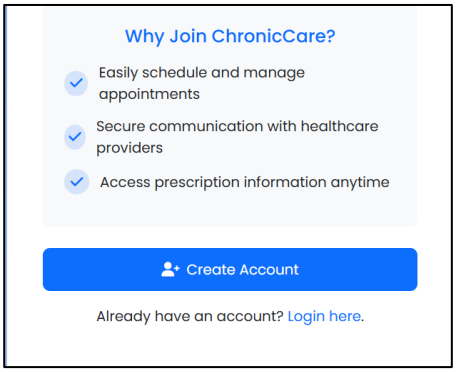


Figure 18 – Register section

All passwords stored in the User table are fully encrypted following the functionality from BCrypt password encoder.

id	username	password	role
1	johncollins	$2a$10$.Gd3lkdBcjBK8BdB78GxtORK1iIa2mGdy...	PATIENT
3	frankdoe	$2a$10$Q/Si9UaoId3R3XLbzHAOJe0UouzgpuPt...	PATIENT
5	frankfelix	$2a$10$6nr3EilOxMg8.ArEhXgpYudZNXW0kylU/...	PATIENT
6	yayoikusama	$2a$10$LSadi8v7m.j9P8umR24pHOxif8pzUegdr...	PATIENT
7	bensmith	$2a$10$/2/foyxtgKSJi6MnadxUP.ln0xH/meJLZ...	DOCTOR

Figure 19 – Password encryption

5.2.2 Dashboard Implementation

5.2.2.1 Doctor Dashboard

Upon successful login, doctors are redirected to their dashboard home page, with accessibility features, a dashboard header with branding and basic doctor information, and buttons for the appointment, message and prescription management features.

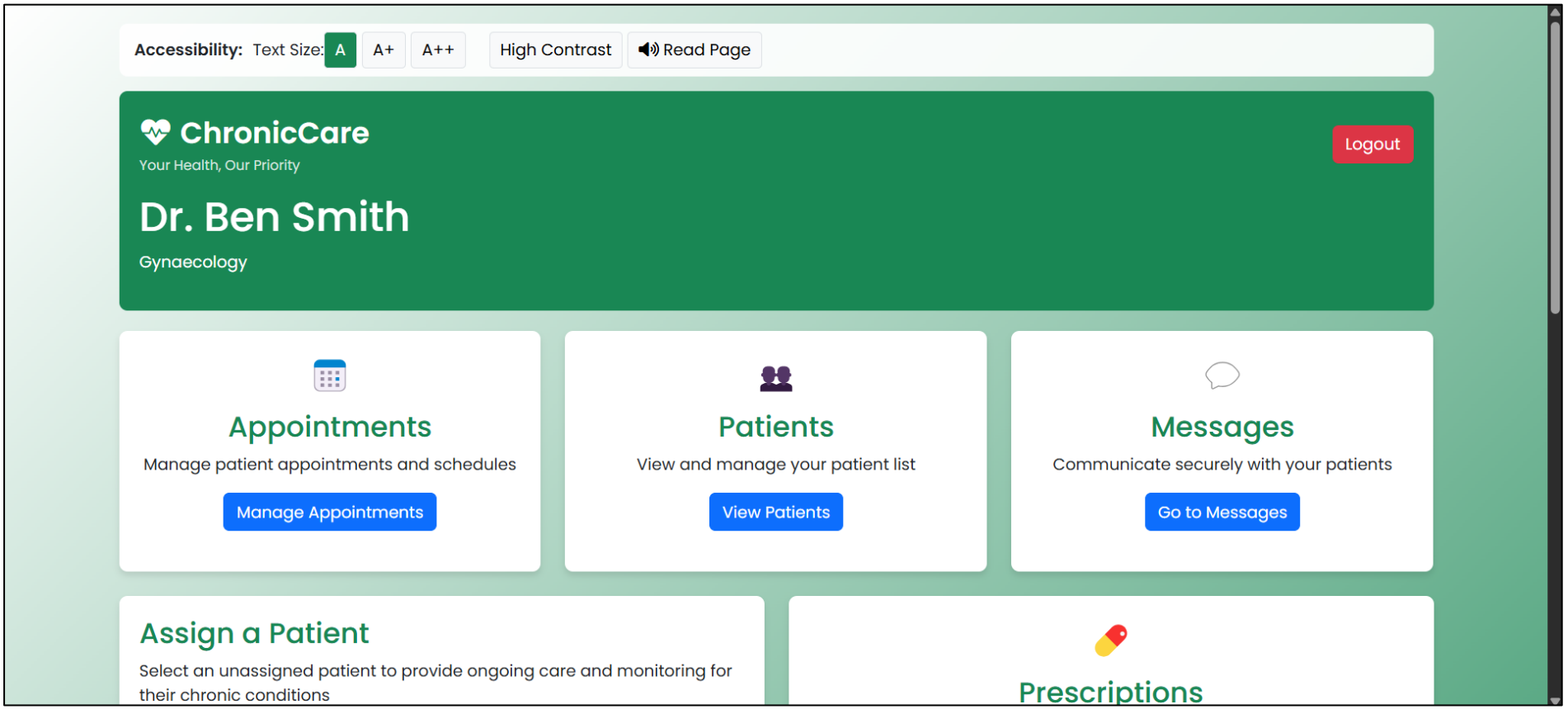


Figure 20 – Doctor dashboard

The dashboard also has a feature for doctors to assign an unassigned patient into their care. The dashboard then updates after the patient is assigned.

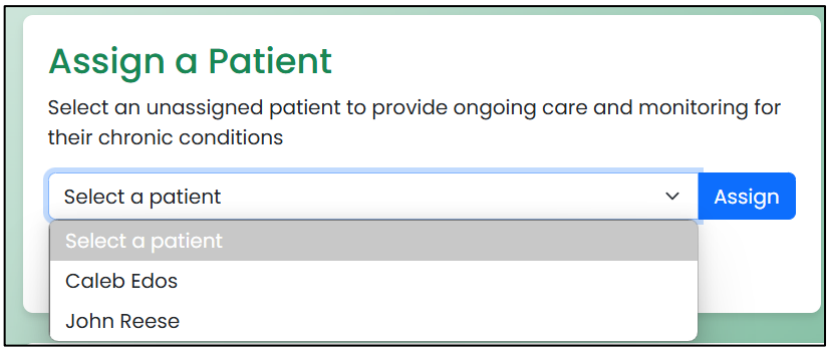


Figure 21 – "Assign a patient" feature

The prescription section for the dashboard is more functional, with more buttons to link to the separate features of prescription management. This is because unlike the other features, the prescription feature is segmented into different pages.

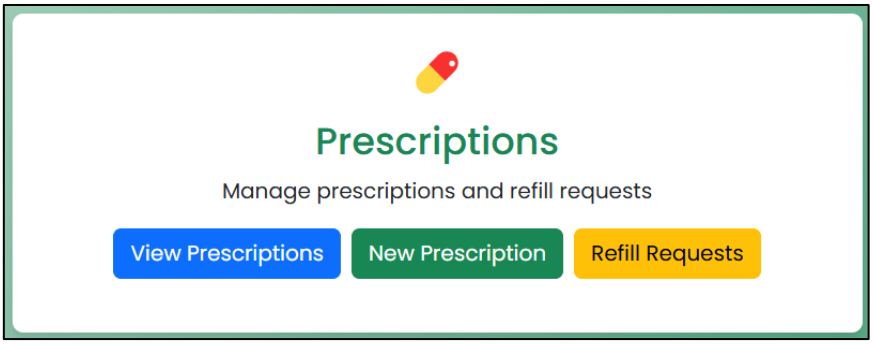


Figure 22 – Prescription feature

The dashboard also contains a "Recent Updates" feature, that shows a brief summary of all recent action taken by their patients, along with buttons to take further action directly from the update. This makes it easier for the doctor to take action quickly [51], further enhancing care management [52].

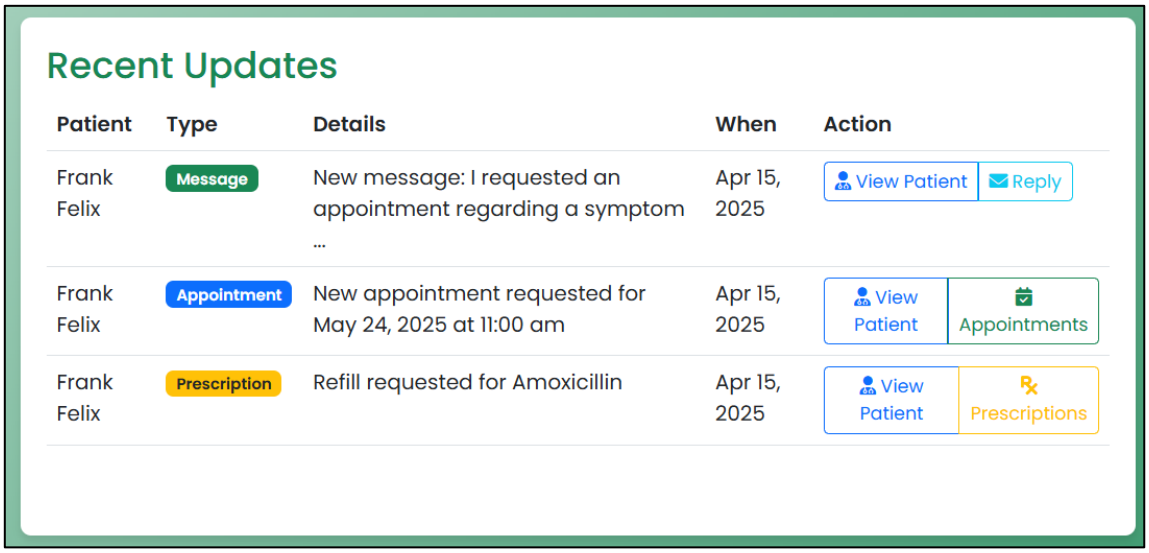


Figure 23 – "Recent Updates" feature

The “View Patients” button takes the doctor to the “Your Patients” section, that shows all their current assigned patients with basic information. Doctors can then view the health record and more in-depth information about the patient through the “View Details” button.

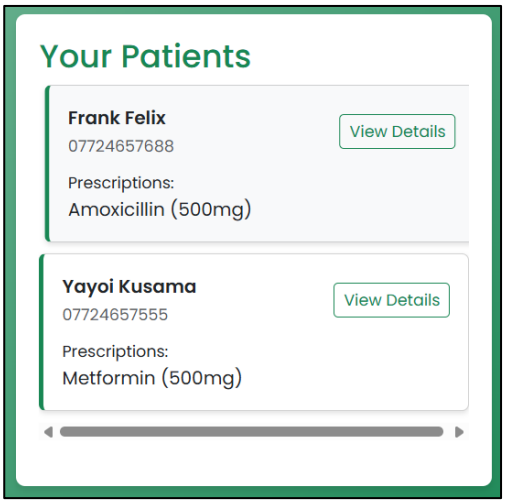


Figure 24 – “Recent Updates” feature

The “View Details” button then accesses the patient details page, allowing them to view the patient’s health record, and view a more comprehensive analysis and summary of the patient’s health data and information.

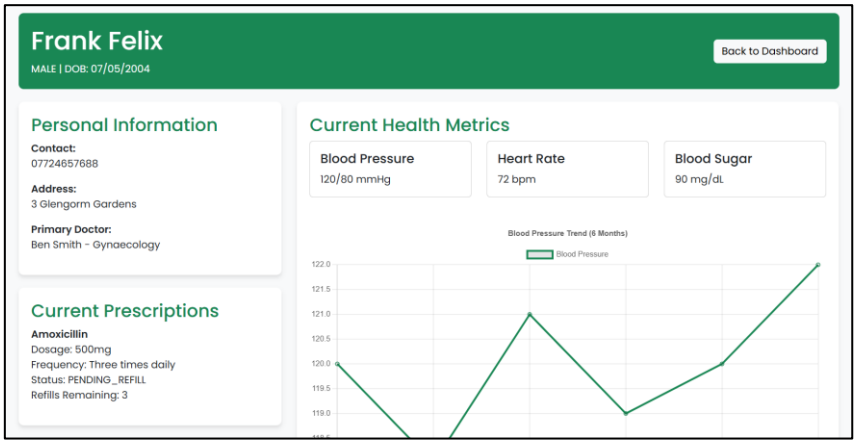


Figure 25 – Patient details page, showing the patient’s health record and related information

Lastly, the dashboard shows the doctor a “Patient Statistics” section, informing the doctor about their total number of patients, notifying them about any patient’s who currently require attention, or any pending appointments.

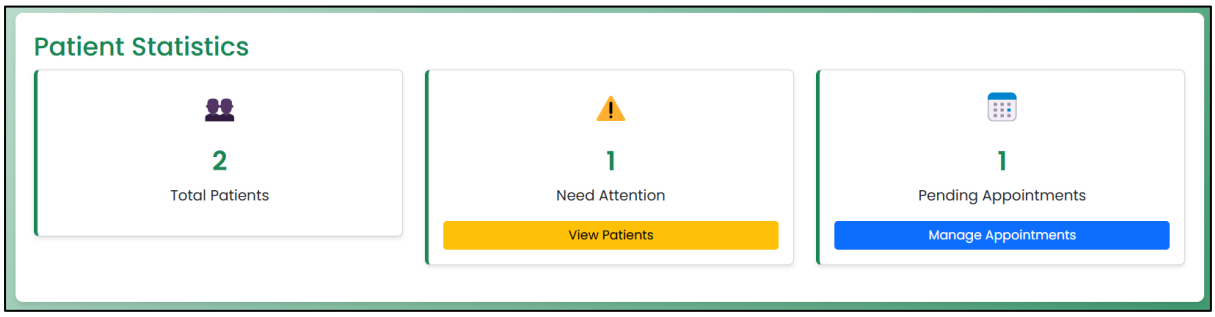


Figure 26 – “Patient Statistics” section

5.2.2.2 Patient Dashboard

Upon successful login, patients are redirected to their dashboard home page, with accessibility features, a dashboard header with branding and their assigned doctor's name, and buttons for the appointment, message and prescription management features.

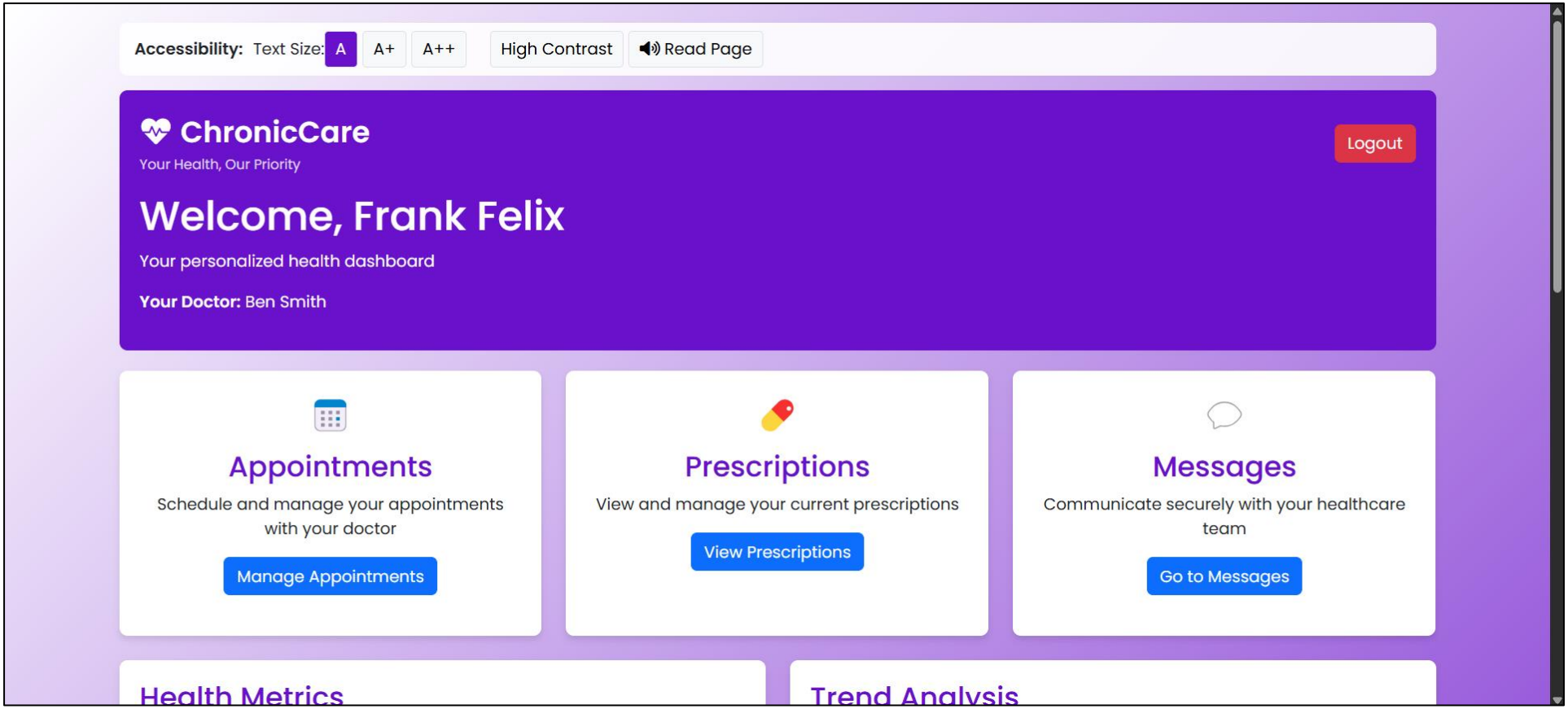


Figure 27 – Patient dashboard

The dashboard also has a “Health Metrics” section, that shows a personalised graph for each patient to visualise the trends in their health.

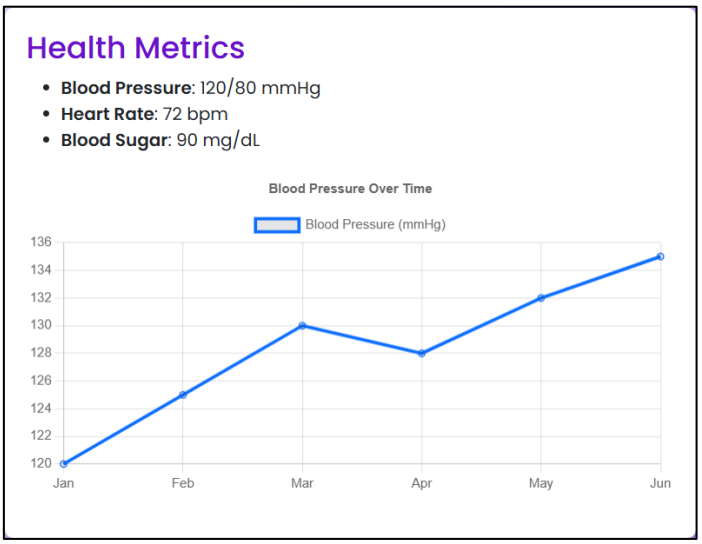


Figure 28 – Health metrics feature

The “Trend Analysis” section then provides a summary and analysis of the trends shown in the health metrics graph.

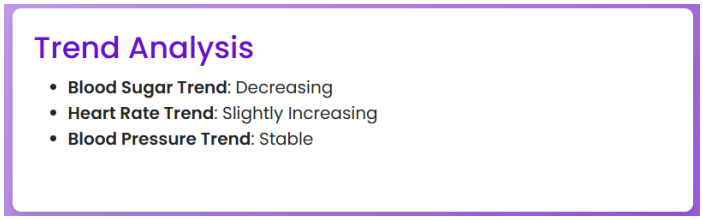


Figure 29 – Trend analysis section

The “Health Assistant” feature provides an interface where the patient can chat with a bot, that has a set of AI-generated responses. This bot provides instant responses, easily offering quick healthcare tips, increasing efficiency and leading to better health outcomes [53].

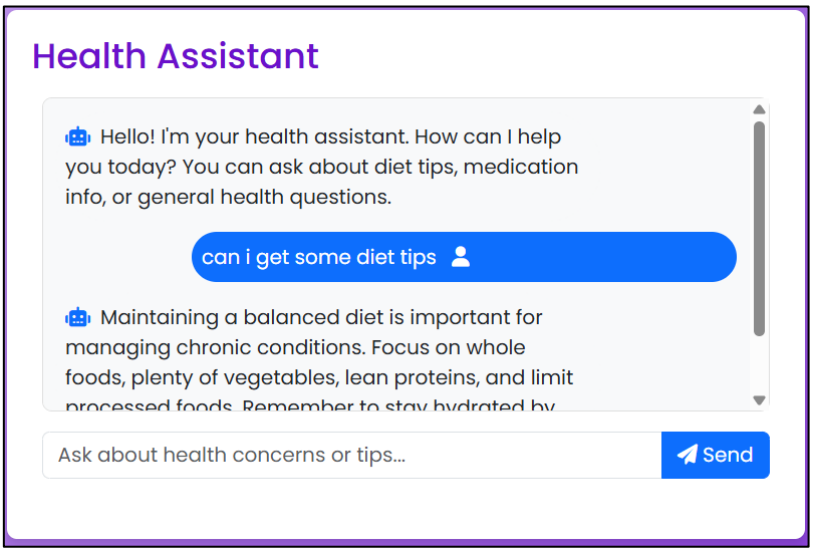


Figure 30 – Health assistant feature

The “Medication Reminder” feature allows patients to set reminders to take their assigned prescriptions.

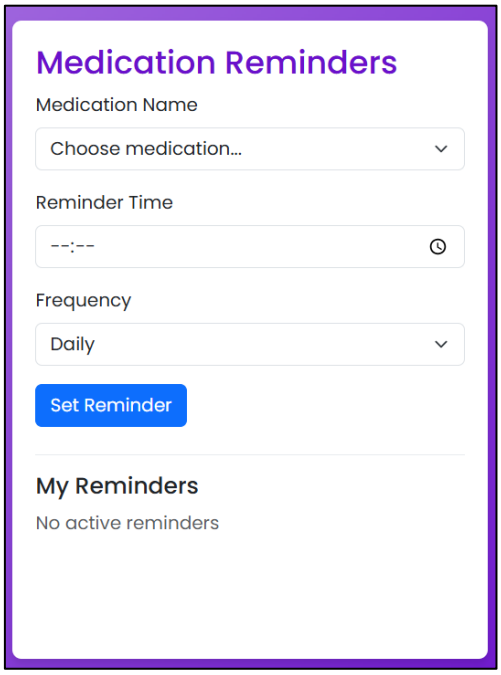


Figure 31 – Medication reminder feature

After a reminder has been set, the patient receives a success message and a notification, as well as the reminder at the bottom of the section.

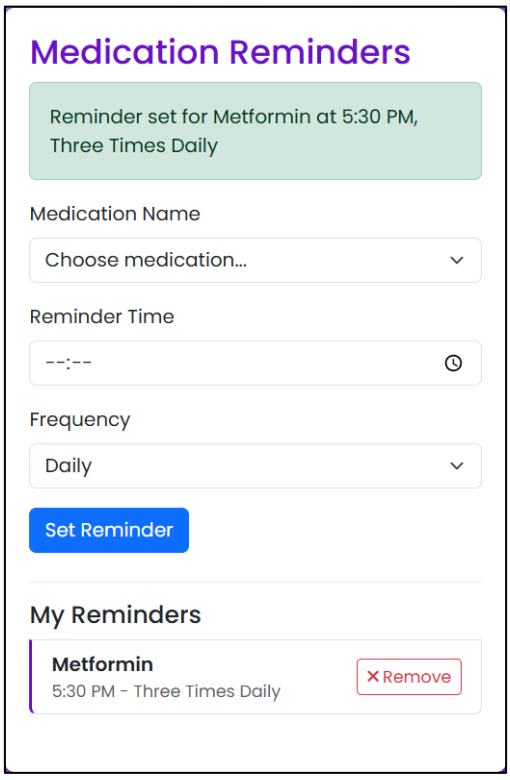


Figure 32 – Successful medication reminder set

The “Prescription Guide” feature allows the patient to choose a medication, and view information regarding how to take their prescription.

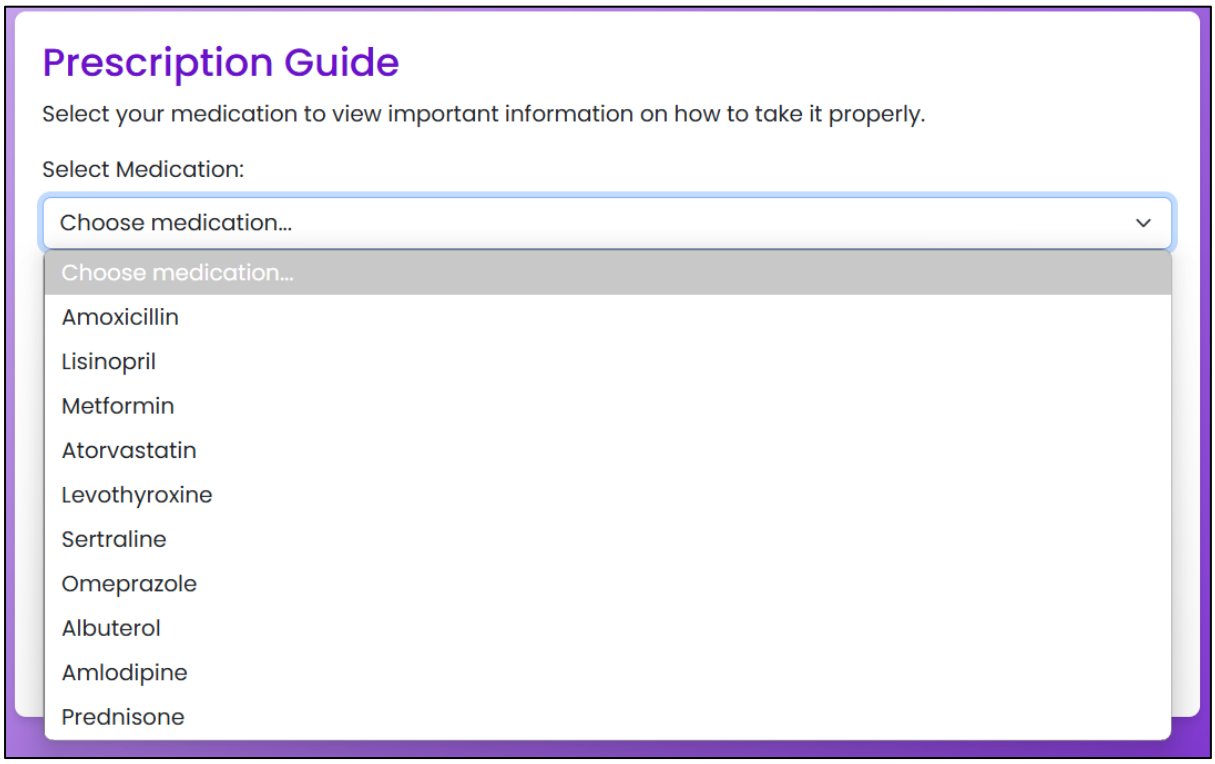


Figure 33 – Set list of medications in the prescription guide feature

After selection, vital information regarding the medication is displayed, and instructions on how to take the medication are presented.

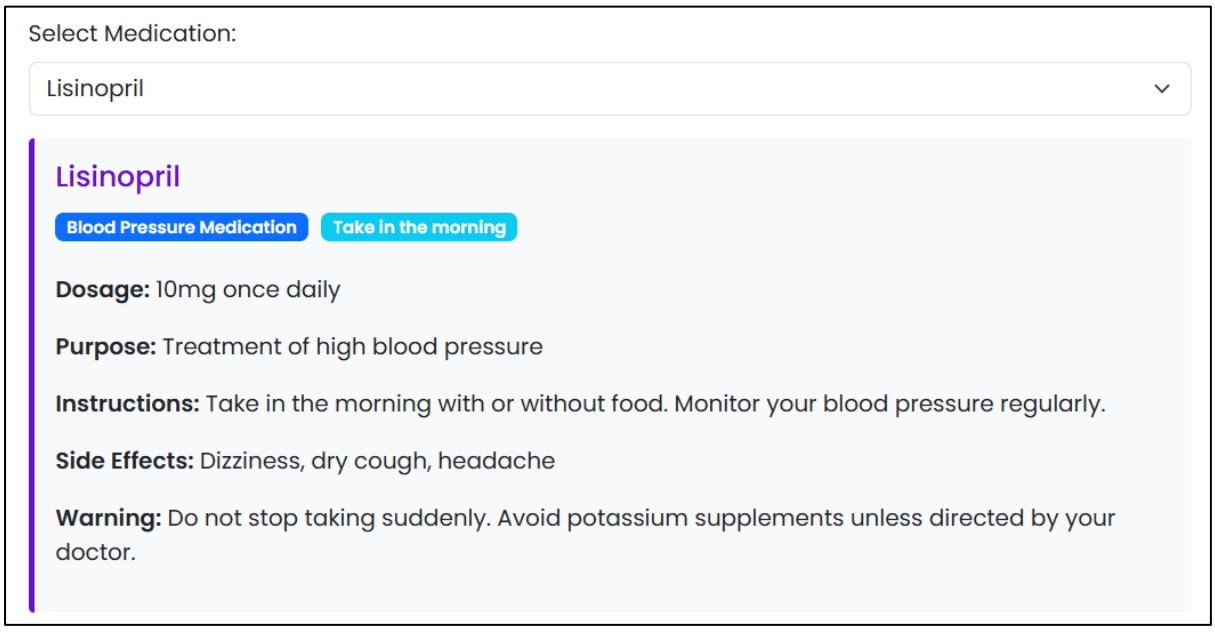


Figure 34 – Vital information and instructions for a medication in the prescription guide feature

5.2.3 Appointment Management Implementation

The “Manage Appointments” button on the patient dashboard, redirects the patient to the main appointment system interface.

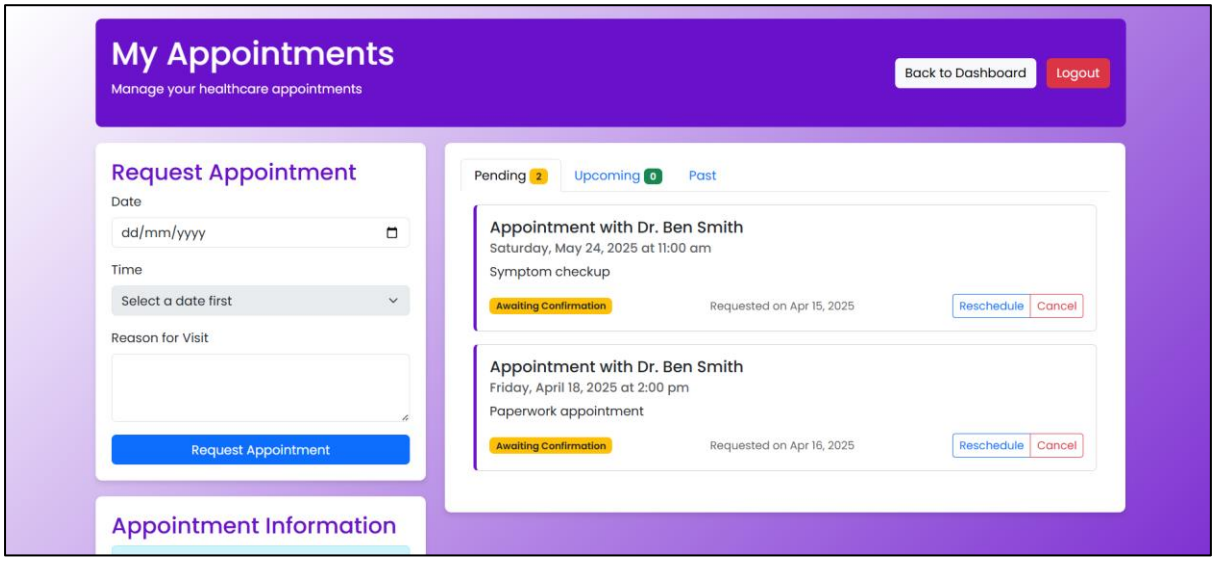


Figure 35 – Patient appointment system

The “Appointment Information” section then gives the user directions on how to book, reschedule or cancel an appointment, and how the booking process works.

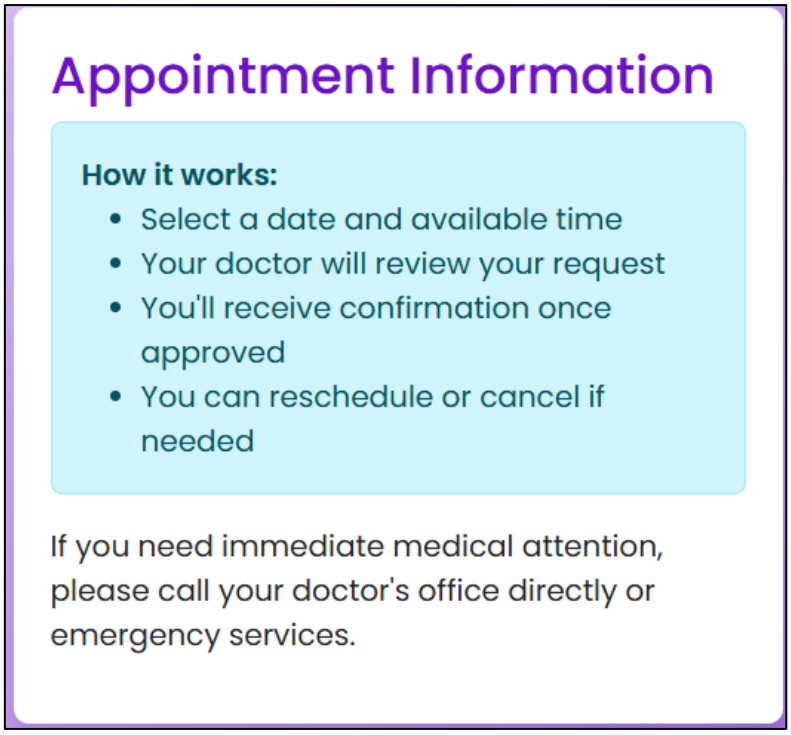


Figure 36 – Appointment information section

The patient can also view more tabs to show the details of any past and upcoming appointments.

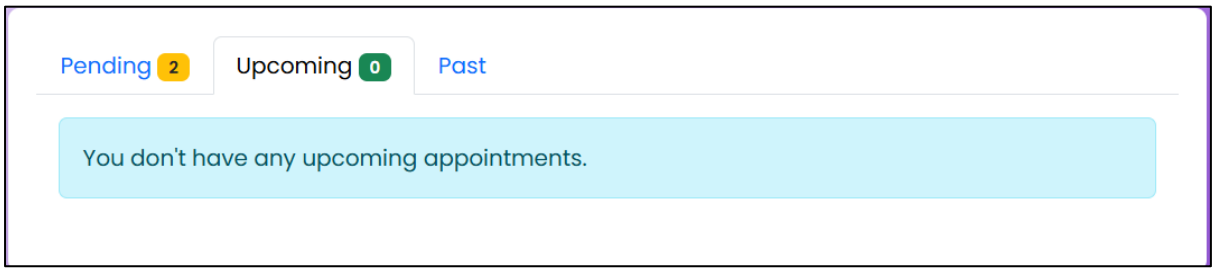


Figure 37 – Upcoming appointments tab

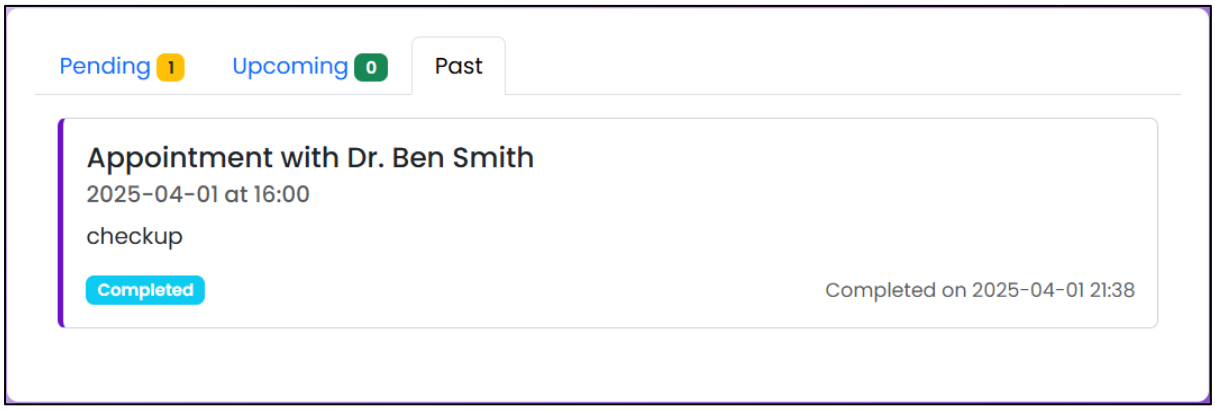


Figure 38 – Past appointments tab

The “Request Appointment” feature then allows the patient to request an appointment with their assigned doctor, choosing a date, time and giving a reason for the visit.

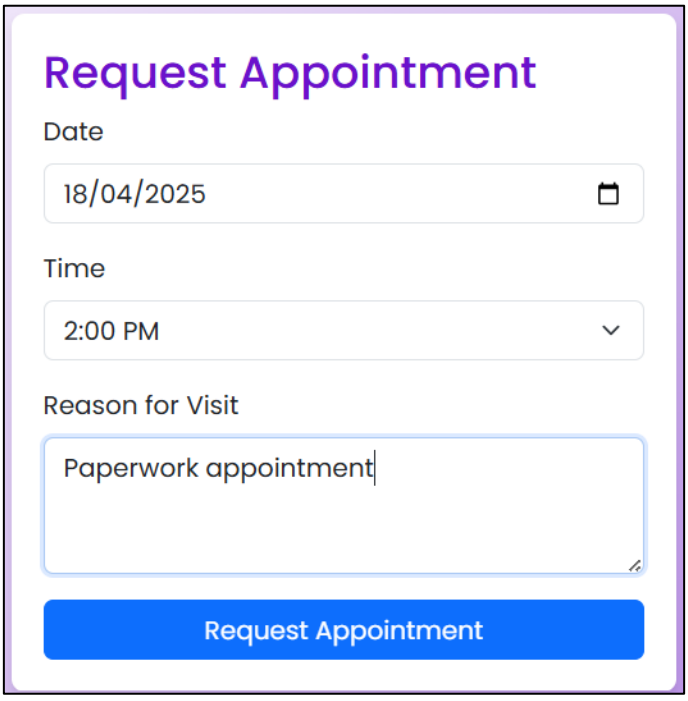


Figure 39 – Request appointment feature

A notification is then showed to the patient, confirming their appointment request.

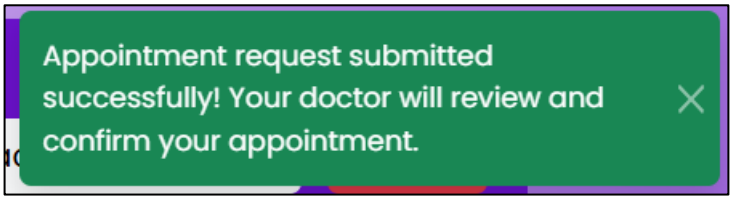


Figure 40 – Appointment request notification

The pending request is then shown to the doctor, on their own respective appointment system interface.

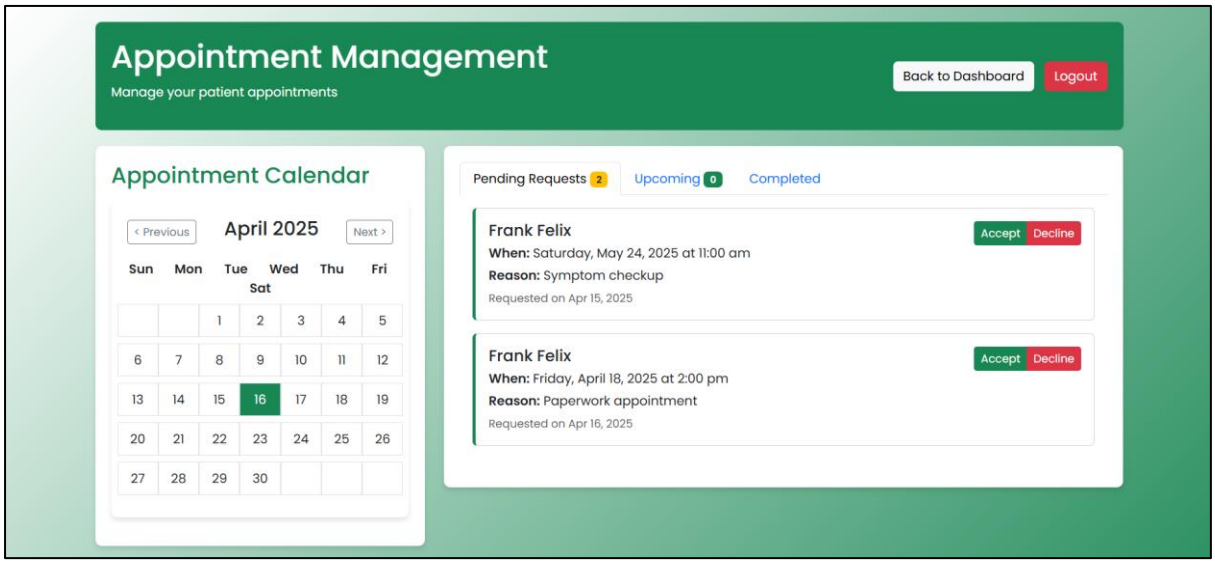
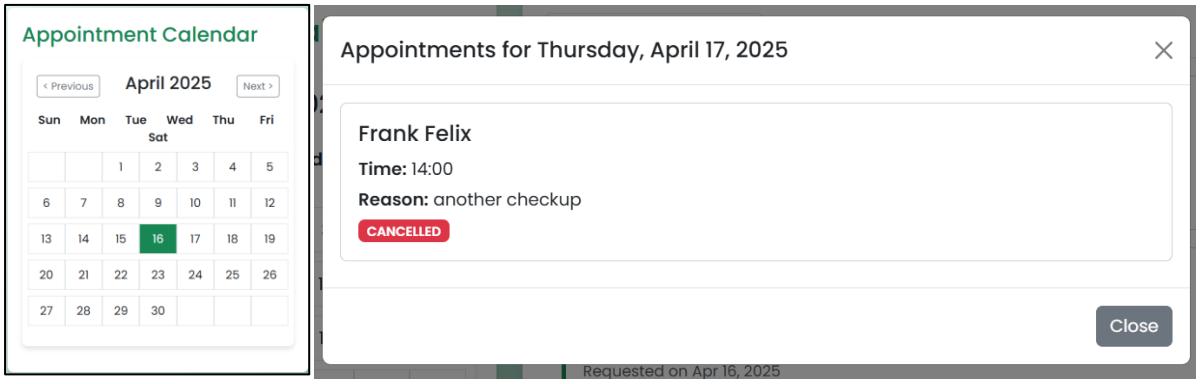


Figure 41 – Doctor appointment system

The “Appointment Calendar” feature allows doctors to view a calendar and click past dates that show information of past appointments, or future dates for future appointments.



Figures 42 and 43 – Appointment calendar system showing the details for April 17th

The “Appointment Calendar” feature shows the doctor the details for any appointments they have scheduled for the current day, if there are any.

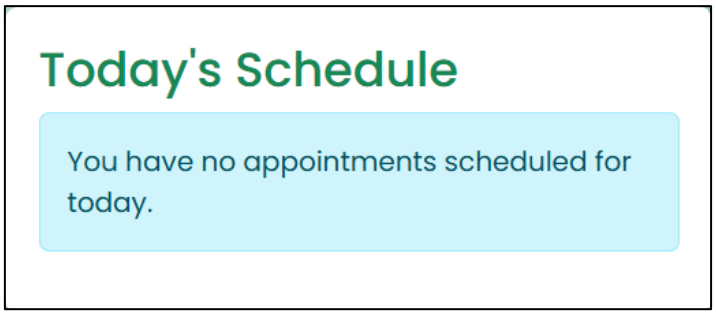


Figure 44 – “Today’s schedule” feature

A dialog box pops up when a doctor clicks “Accept Appointment” allowing confirmation.

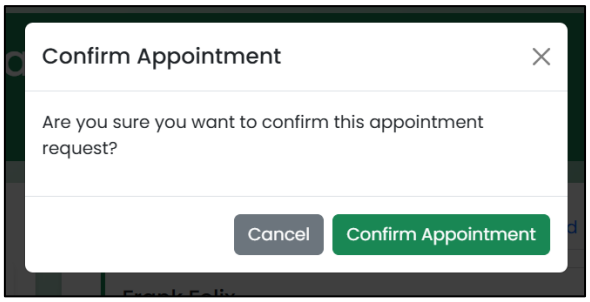
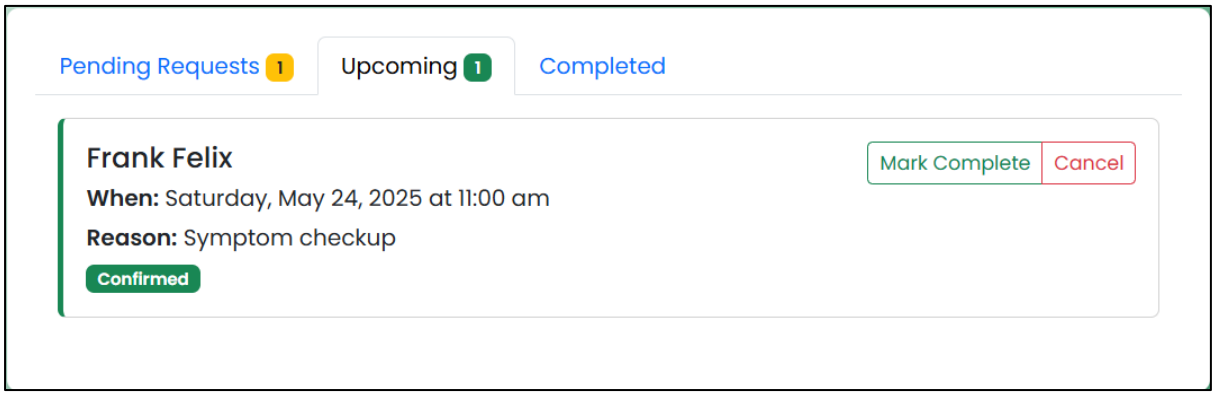


Figure 45 – Confirm appointment dialog box

After confirming the appointment, the appointment is then shown in the upcoming appointments tab for both the doctor and patient.



Figures 46 and 47 – Confirmed scheduled appointment appearing for both the patient and doctor

The “Mark Complete” button prompts a dialog box to pop up, allowing the doctor to mark the appointment as completed, also allowing them to enter any important notes.

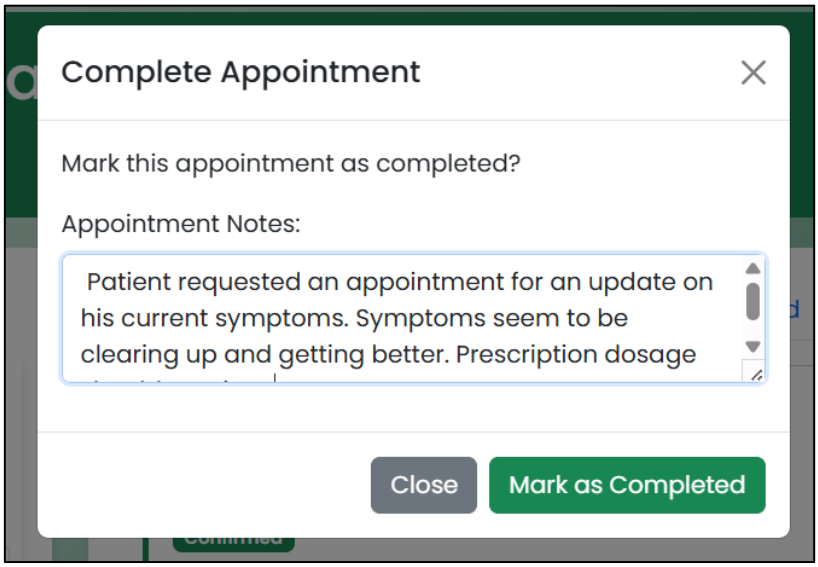
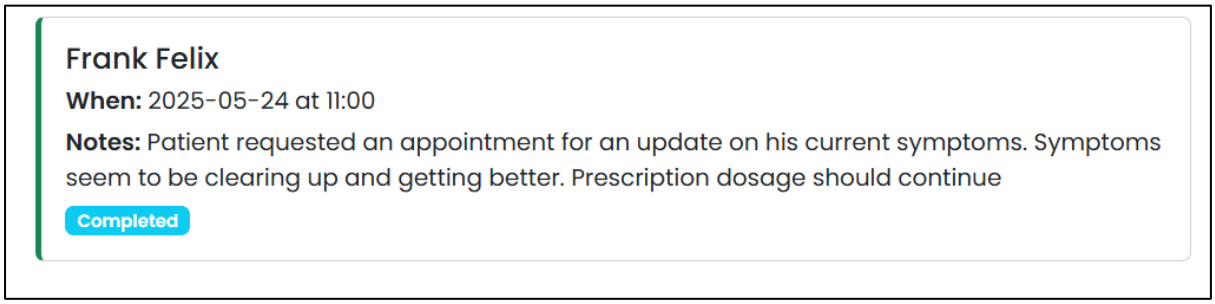


Figure 48 – Complete appointment dialog box

After completing the appointment, the appointment is then shown in the past/completed appointments tabs for both the doctor and patient.



Figures 49 and 50 – Completed appointment appearing for both the patient and doctor

The “Reschedule Appointment” button allows the patient to request to reschedule an appointment for a new date and time.

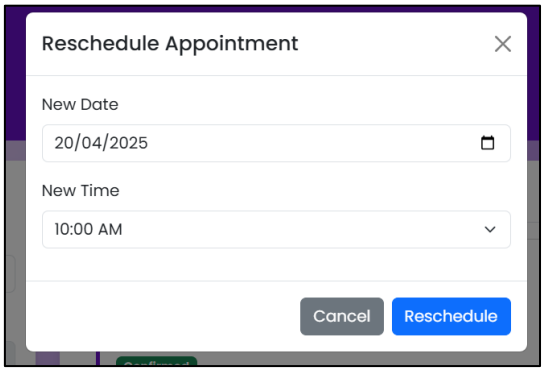


Figure 51 – Reschedule appointment dialog box

After rescheduling the appointment, the appointment becomes a request again.

Figure 52 – Rescheduled appointment as a request

The “Cancel Appointment” button allows a patient or doctor to cancel with a reason.

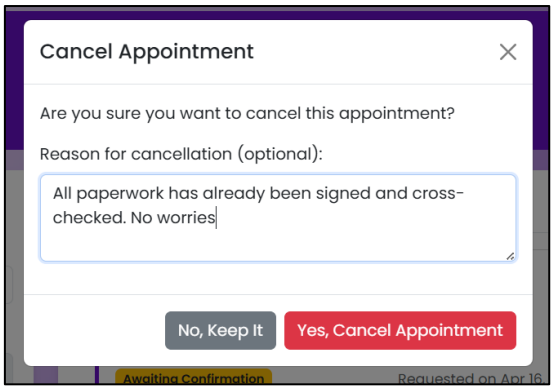


Figure 53 – Cancel appointment dialog box

After cancelling, the appointment leaves the upcoming appointments tab.

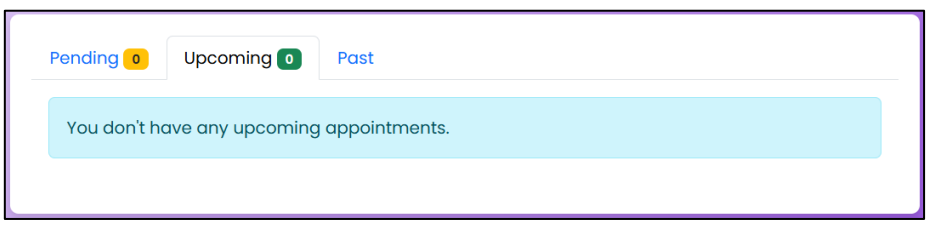


Figure 54 – Empty upcoming appointments tab after a cancelled appointment

5.2.4 Communication System Implementation

The “Go To Messages” button on each dashboard, redirects the user to a messaging interface tailored to them, showing a history of the messages **only they** have sent and received.

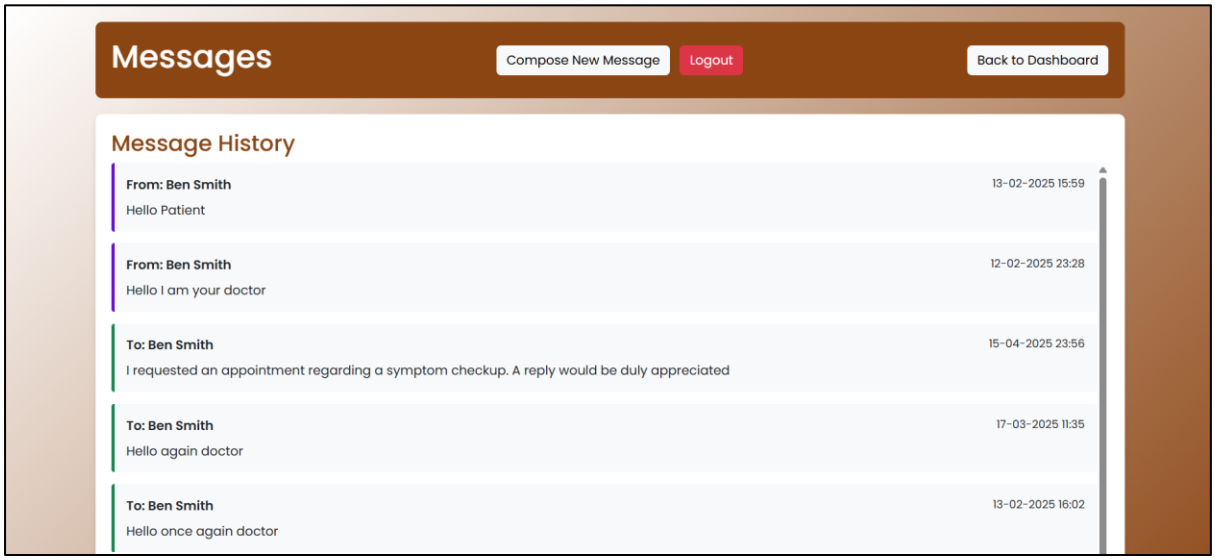


Figure 55 – Messaging system for patient Frank Felix

The “Compose New Message” button, then leads the user to a page with a form, that allows them to send a message.

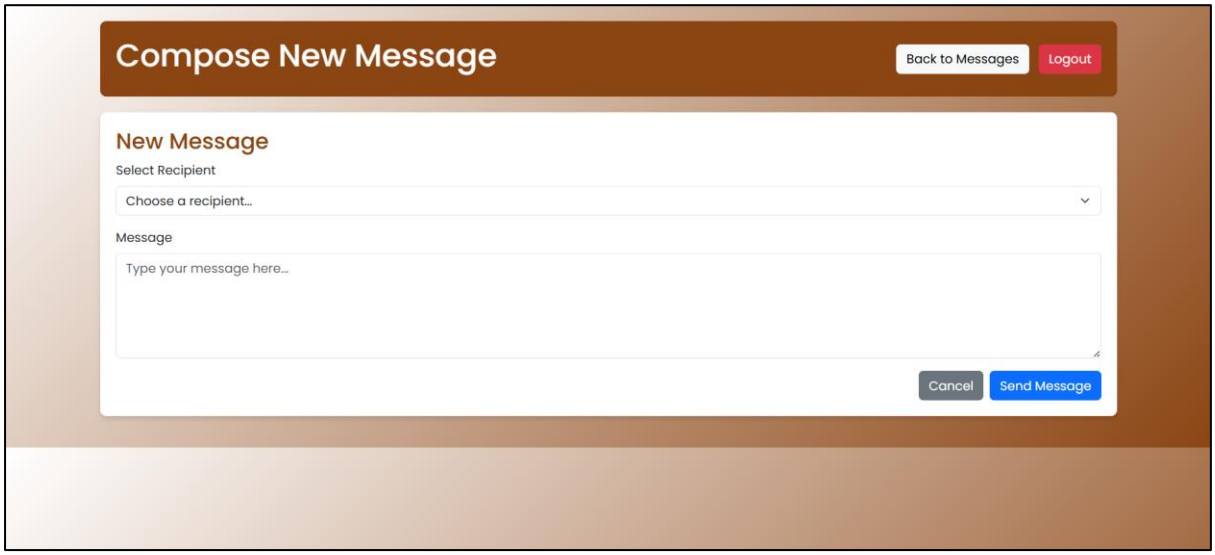


Figure 56 – Compose new message page

The user can then choose from a list of registered users on the system to send their message to.

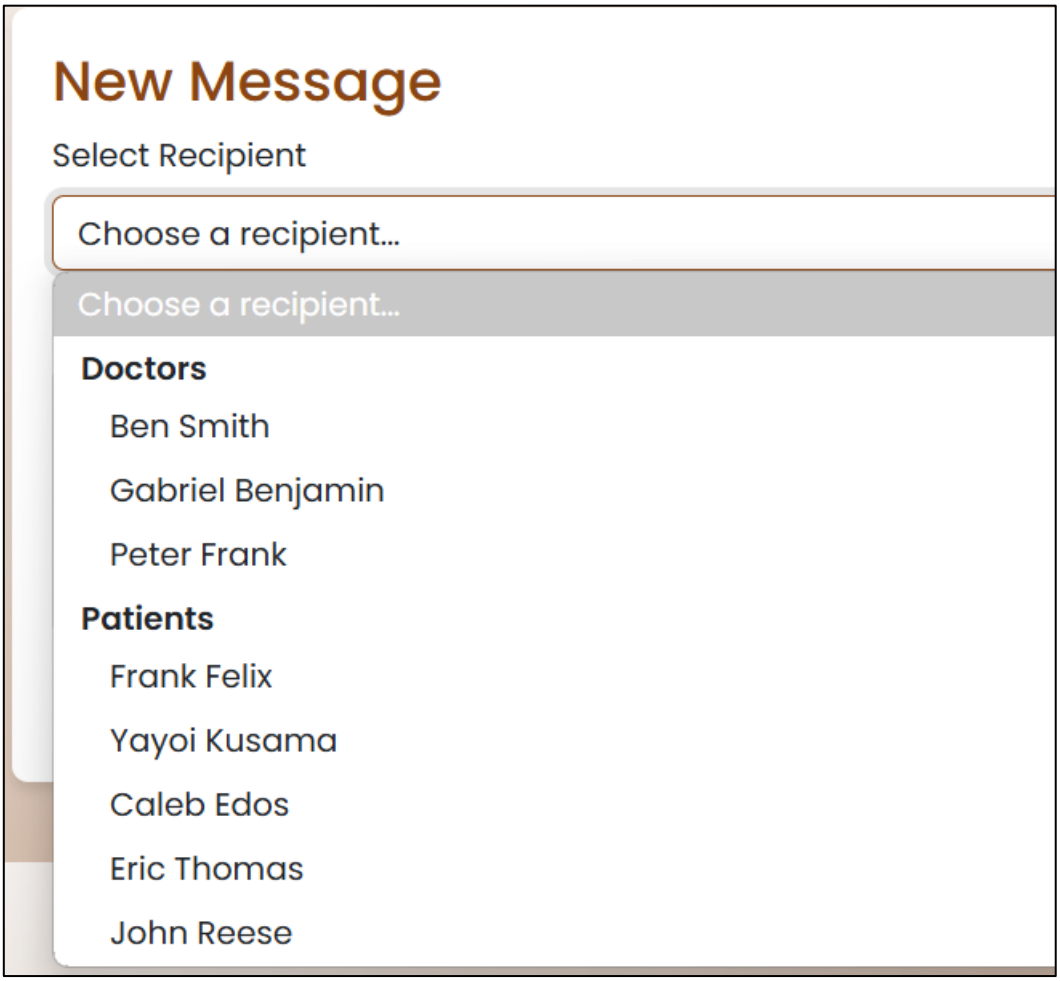


Figure 57 – Recipient list for messages

The user can then compose their message in the content box, and send.

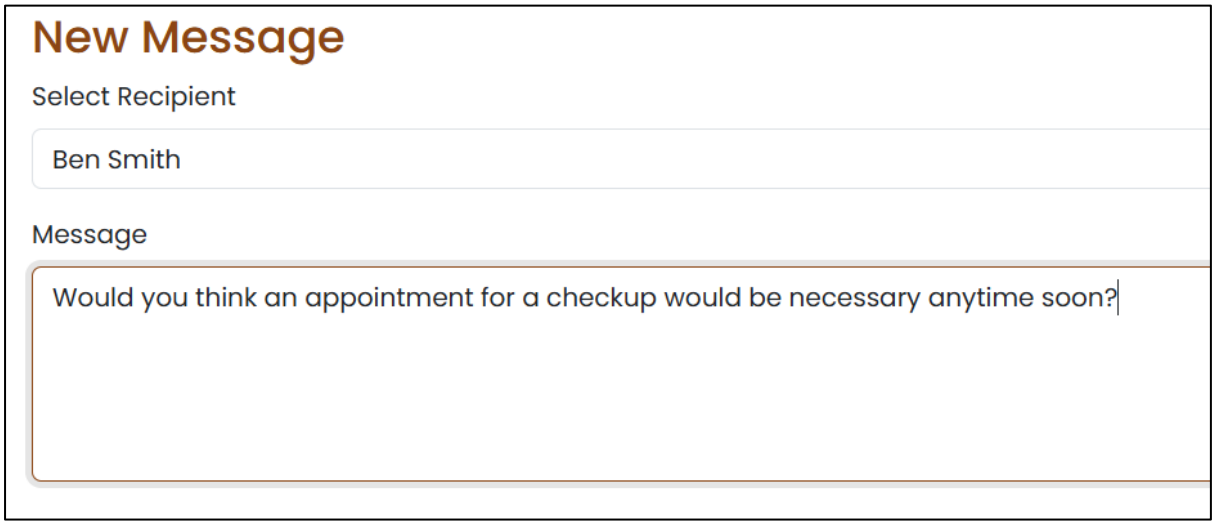


Figure 58 – Message currently being composed by patient Frank Felix, to his doctor Ben Smith

After receiving the message, a doctor can briefly see it in his “Recent Updates” section.

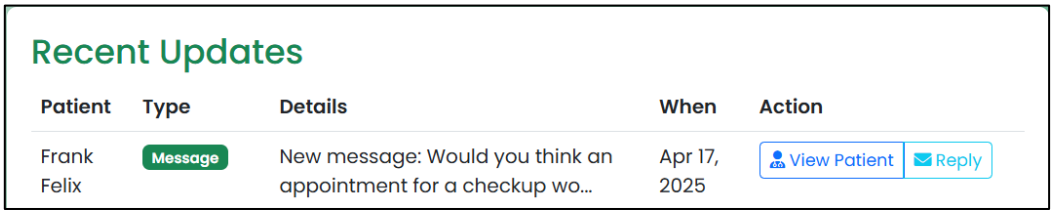


Figure 59 – “Recent Updates” section for doctor Ben Smith, showing the most recent received message

The doctor can then see a history of the messages **only they** have sent and received, with the most recent received message being the first.

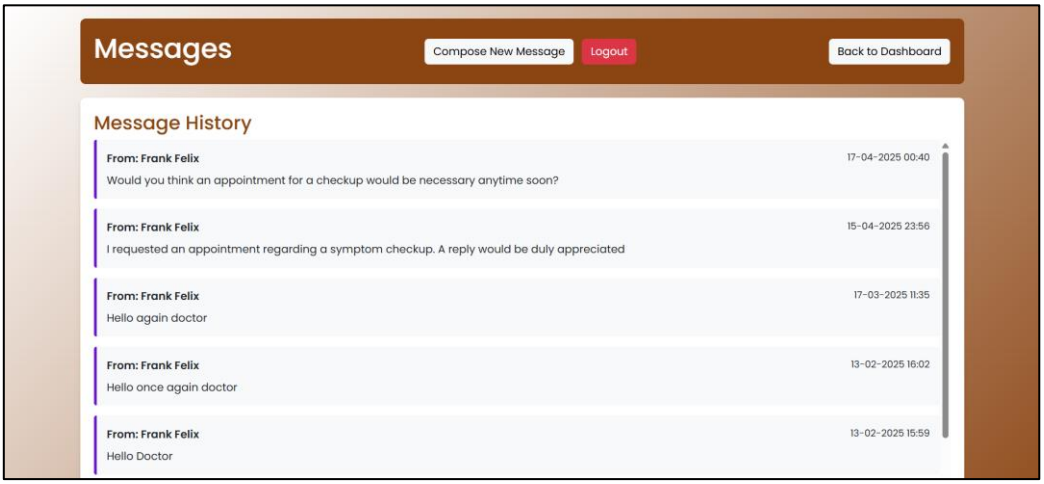


Figure 60 – Messaging system for doctor Ben Smith

5.2.5 Prescription Management Implementation

The “View Prescriptions” button on the doctor dashboard, redirects them to their main prescription management interface, allowing them to see the information for all their created prescriptions for their assigned patients.

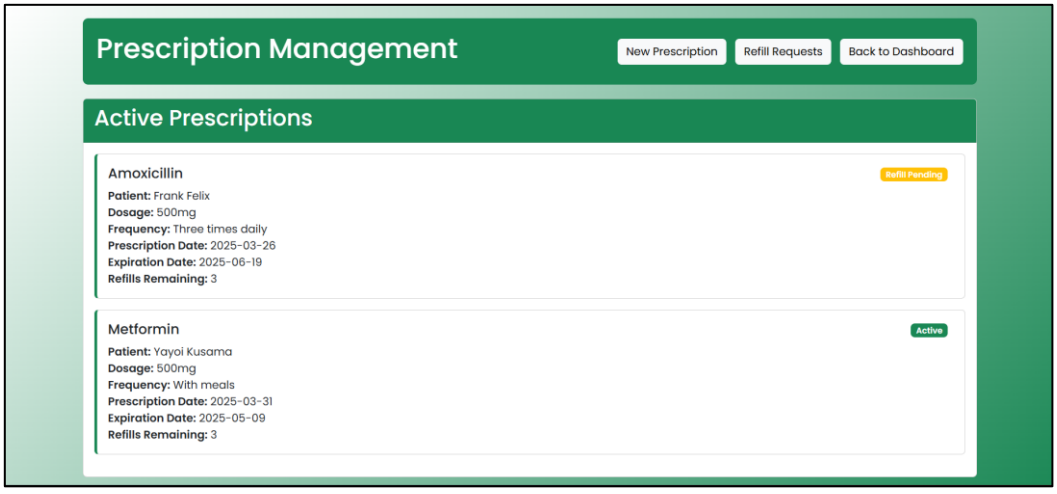


Figure 61 – Prescription system for doctor Ben Smith

The “Create Prescription” button redirects the doctor to a page with a form, allowing them to select a patient from a list of **only** their assigned patients, choose the medication, dosage, frequency, expiration date, refill amount, and include additional instructions.

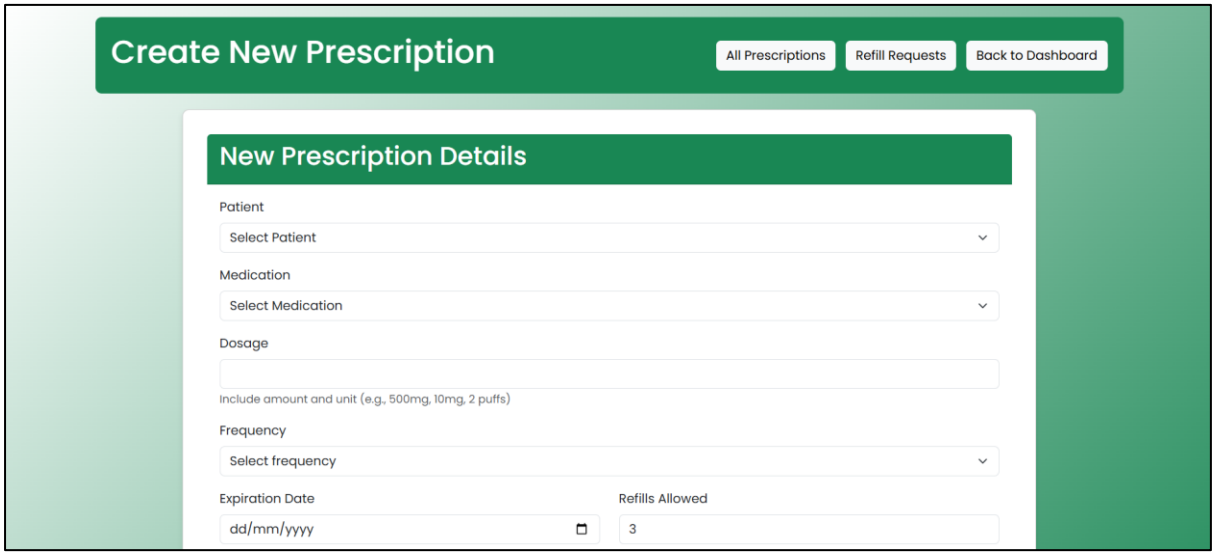


Figure 62 – “Create new prescription” page

The form contains functions to fill out the form based on the instructions from the prescription guide if necessary, making the prescription creation process easier.

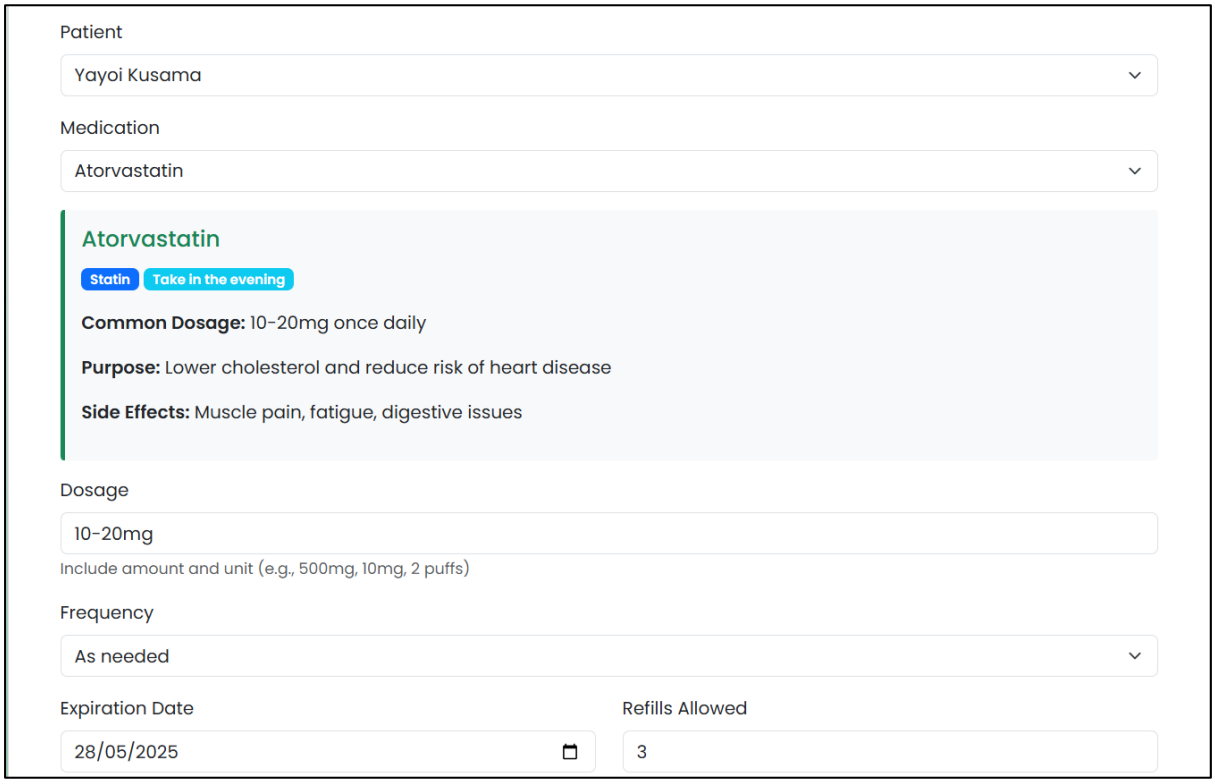


Figure 63 – Filled form for a new prescription

After the prescription has been created, it shows as an active prescription.

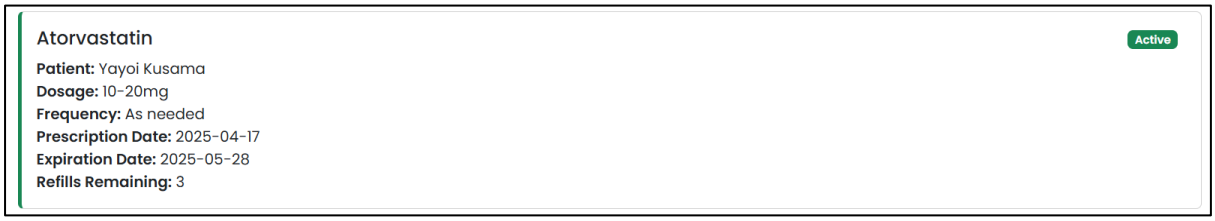


Figure 64 – Active prescription shown to the doctor after creation

The “View Prescriptions” button on the patient dashboard, redirects them to their main prescription management interface, allowing them to see the information for all their assigned prescriptions by their doctor, and request refills for any active prescriptions. The prescription that was just created also shows for the patient.

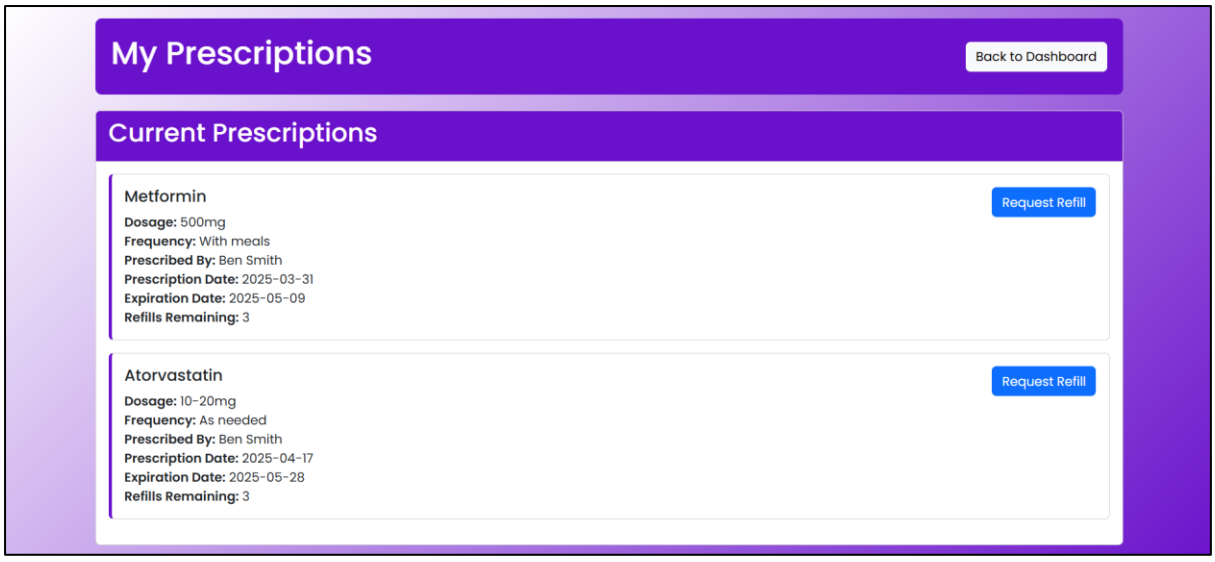


Figure 65 – Filled form for a new prescription

After the patient clicks the “Request Refill” button, the status of the prescription is updated to “Refill Pending”, and becomes a refill request for the doctor.

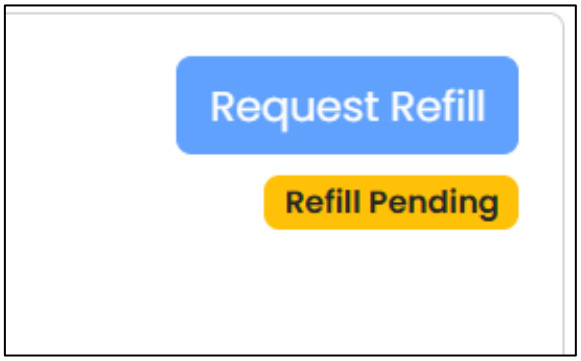


Figure 66 – Updated “Refill Pending” status for a current prescription

After receiving the refill request, a doctor can briefly see the details of it in his “Recent Updates” section, with quick action buttons to take the doctor to their prescriptions.

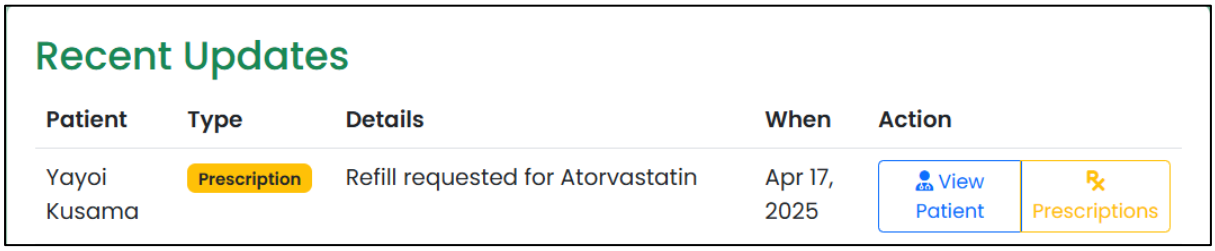


Figure 67 – The doctor’s “Recent Updates” section, showing the most recent refill request

The doctor’s “Refill Requests” page then shows all the current refill requests, with options to either approve or deny the requests.

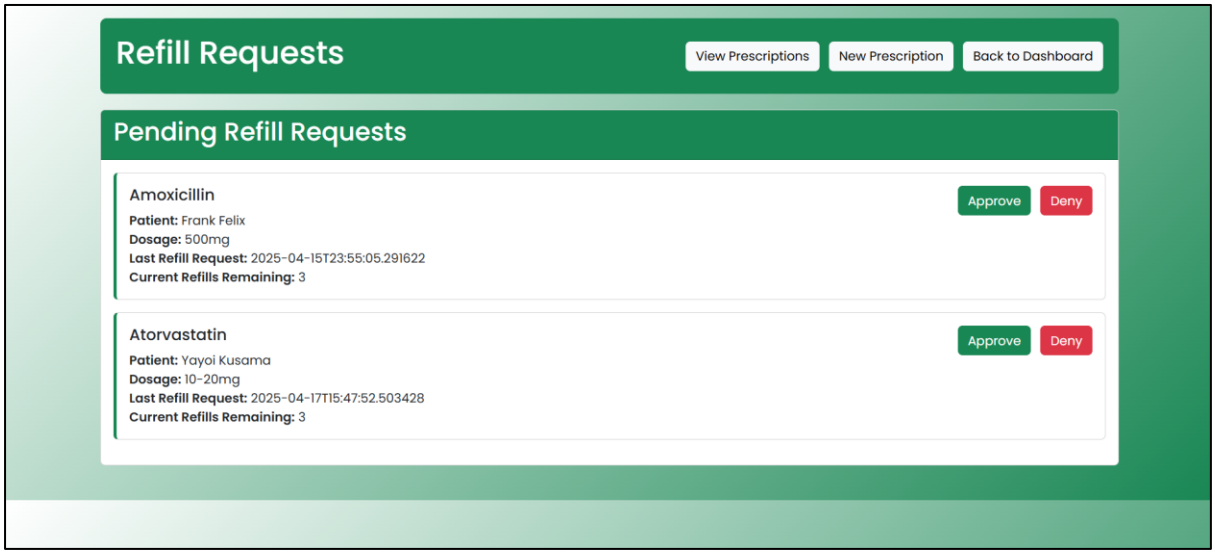


Figure 68 – Refill requests page

After accepting or denying all pending requests, the request tab empties.

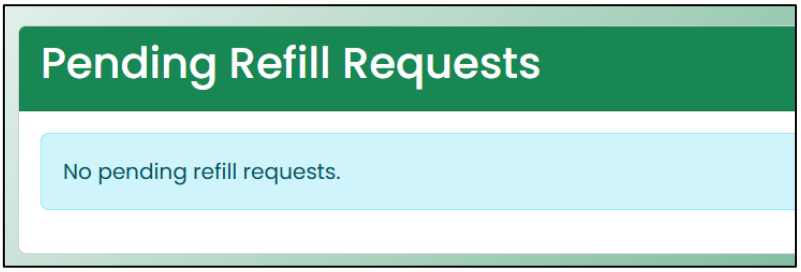


Figure 69 – Empty refill requests tab

5.3 Testing and Results

I implemented a very rigid and structured testing framework for my completed software system. The testing stage is crucial in the software development cycle, as it ensures the system is free of bugs and defects, improving the quality and reliability of the software system as a whole [54]. This then improves customer satisfaction, and the chance of scalability of the system in the long term [55].

5.3.1 Unit Testing

Unit testing is crucial, as it ensures that each individual vital component that makes the system is working properly [56].

For my unit testing, I used **JUnit 4** as the main testing framework in Java, **Mockito** to avoid database operations and focus on the functionality of the code alone, and **Spring Boot Starter Test** for general component testing in Spring Boot.

Some of the unit tests are AI-generated. This is because it is shown in unit testing that not only will AI apply a high level of consistency and thoroughness when testing, it identifies and tests against cases and situations that humans may overlook, preventing bugs making it through to the final finished product [57].

The 4 main files for unit testing in my system are:

- **HealthSystemApplicationTests**: This file will include basic tests for the core functionalities of the system.
- **HealthSystemServiceTests**: This file will include tests for all the service classes in the system.
- **HealthSystemControllerTests**: This file will include tests for the controller classes in the system.
- **HealthSystemSecurityTests**: This file will include tests for the security configuration of the system.

The summary of all tests, their description and results are as below:

No.	Test Name	Description	Result
HealthSystemApplicationTests			
1	contextLoads	Tests that core Spring boot components load with no errors	Pass
2	passwordEncoder_ EncodesAndVerifiesPasswords	Tests that password encryption and verification works	Pass

3	coreServiceComponents_ FunctionProperly	Tests that the core “get” functionalities of services work	Pass
4	patientHealthMetrics_ AreAvailable	Tests that health metrics are returned	Pass
5	patientTrendAnalysis_IsAvailable	Tests that correct health trend analysis is returned	Pass
6	modelAssociations_ FunctionCorrectly	Tests that model relationships (doctor manages patient, patient has prescriptions, etc) work	Pass
7	entityInheritance_WorksCorrectly	Tests that patient and doctor inheritance from user class works	Pass
HealthSystemServiceTests			
8	getPatientByUsername_ ReturnsCorrectPatient	Tests that the patient service correctly retrieves patient users by username	Pass
9	getHealthMetrics_ ReturnsNonEmptyMap	Tests that the patient service correctly returns health metrics for a patient	Pass
10	getDoctorByUsername_ ReturnsCorrectDoctor	Tests that the doctor service correctly retrieves doctor users by username	Pass
11	getPatientStatistics_ ReturnsValidStatistics	Tests that the patient statistics feature returns correct information from services	Pass
12	createAppointment_ SetsCorrectStatus	Tests that appointment creation initially sets an appointment to the right status	Pass
13	confirmAppointment_ UpdatesStatus	Tests that appointment confirmation updates an appointment to the right status	Pass
14	createPrescription_ SetsCorrectStatus	Tests that the prescription service properly creates and sets prescriptions for a patient	Pass
15	sendMessage_ CreatesMessageWithTimestamp	Tests that the message service properly creates a message	Pass
HealthSystemControllerTests			
16	loginPageShouldBeAccessible	Tests that the auth controller allows unauthenticated users to access the login page	Pass

17	registerPageShouldBeAccessible	Tests that the auth controller allows unauthenticated users to access the register page	Pass
18	dashboard_ ReturnsCorrectViewAndModel	Tests that the doctor and patients controllers correctly return the unique dashboards and user attributes	Pass
19	viewPatientDetails_ ReturnsCorrectViewAndModel	Tests that the doctor controller allows the doctor to access their assigned patient details	Pass
HealthSystemSecurityTests			
20	publicEndpoints_ ShouldBeAccessible WithoutAuthentication	Tests that unauthenticated users can still access public endpoints and pages	Pass
21	protectedEndpoints_ ShouldRedirectUnauthenticated UsersToLogin	Tests that doctor and patient endpoints and pages can only be accessed after authentication	Pass
22	patient_ ShouldNotHaveAccess ToDoctorEndpoints	Tests that patients can not have access to doctor endpoints and pages	Pass
23	doctor_ ShouldNotHaveAccess ToPatientEndpoints	Tests that doctors can not have access to patient endpoints and pages	Pass

All 23 unit tests passed successfully, indicating the logic and functionality of the individual components of my system are robust, accurate and reliable.

5.3.2 System Testing

To ensure the complete system works correctly and accurately as a whole [58], I performed a simple set of system tests. These tests involved ensuring the user interface works as expected, and testing the end-to-end workflow of the system.

This testing process involved me running multiple iterations of the login and register process, filling out forms across the system with various data types each time, and navigating round the features and buttons of the system as each type of user. This focused on not only testing the functionality of the user interface, but also testing the validation techniques, the data flow of the system, and the general end-to-end work flow of the system as all components interact together.

This testing process actually identified a bug in the registration process, where all user details were being stored in the database with a comma at the end or start of the data. For example, a name would be saved as ",Ben Smith" instead of "Ben Smith". This was due to a coding inaccuracy in the role-toggle functionality on the register page, as empty data was being submitted for the role-specific fields of the role not chosen, causing Spring to separate the empty object data with a comma. This was then fixed by updating the JavaScript to disable unused form fields when switching roles.

5.3.3 User Testing

End-user testing my system is crucial, as it ensures the system meets real-world usability needs and expectations, increasing user satisfaction and ensuring the system is actually relevant towards meeting the needs of its intended users [59].

Due to my limited access to people who actually meet the intended target audience, I extensively and meticulously tested my system with only one end user: an elderly family friend who currently uses the NHS application to manage their chronic illness.

The testing focused on allowing the user to extensively use the system, and give detailed feedback based on the following questions:

- What do you like about the system?
- What do you dislike about the system?
- Are you satisfied with the functionality of the system?
- Is the system useful to you?
- Are there any areas of improvement for the system?

What do you like about the system?

The user immediately expressed their appreciation of the centralised health management dashboard. The ability to view prescriptions, schedule appointments, and communicate with their doctor was found to be very useful. The user liked that they can track their health metrics, and speak to a health assistant if they could not be bothered to contact their doctor for a minor problem or general advice. They also appreciated the design of the user interface, calling it "neat" and "visually appealing", and stating that the color-coding and intuitive navigation made it easy to find what exactly they needed.

What do you dislike about the system?

The user stated that although the system makes booking appointments with their doctor easy and convenient, they cannot log their current symptoms or add notes for things they would want to say during appointments. They stated that if they were

experiencing new symptoms or changes in their condition, they may forget to mention these during appointments or send messages about them.

Are you satisfied with the functionality of the system?

The user stated that they were generally satisfied with the core functionality of the system, saying that it addresses all their essential needs and will generally make their management of their healthcare much easier, but stated there is still room for improvement.

Is the system useful?

The user stated that in practical sense, the system would be extremely useful in managing their healthcare, saying that it addresses the main problem they personally have: lack of communication, and difficulty with navigating through such systems.

Are there any areas of improvement for the system?

The user was generally satisfied with the current features of the system, but stated that a feature where you could log your symptoms and current side-effects would be particularly useful, and would give their doctor more context during appointments. The user also stated that a feature that could give illness-specific information and provide educational resources for research would be extremely useful, to keep them informed and would help them understand their illness and treatments a lot better. Lastly, the user stated that the ability to further customise their own dashboard to fit their own needs would help, as they can then navigate the system how they would want to.

In summary, all stages of testing were crucial, as the robust framework allowed me to identify any bugs and errors with individual system components, ensure the system would work as intended as a whole complete application, and verify that the system was fully useful and relevant towards the end-user target audience.

6. Critical Appraisal and Evaluation

6.1 Critical Analysis

In this section, I will give a thorough analysis of the completed software system and project as a whole, analysing my original objectives for the project, and deducing whether or not these were achieved. This process will also highlight the approaches I took, what went well, what did not go well, and any alternative approaches I could have taken in hindsight for general improvement.

6.1.1 Objectives Evaluation

Gather information from background research

Extensive research was accurately gathered on the history of digital platforms in healthcare during the course of the project. I believe this research not only allowed me to acquire necessary knowledge needed to create a useful system for managing the healthcare of chronic patients, but also made me realise how big of a problem fragmented care and communication in healthcare actually is.

I found research for the project relatively easy, because there are so many articles bringing light to the problem at hand, but not many developers attempting to actually fix the issue. The main research approach I used was typing queries into Google and reading articles related to digital solutions in healthcare, and backing up any crucial information with references. I wouldn't make changes to my research approach in hindsight, as I believe I made the process as accurate and extensive as I could.

Design an interactive web-based application in Spring Boot

One of the main aims of the project was to design and create a centralised web based application in Spring Boot, with distinct functionalities for a patient and doctor, which my system effectively does.

Analysing the finished system thoroughly, the system has been correctly implemented based on all the functional requirements. The system fully supports custom user management, allowing both types of users to securely register and login. The system also has two distinct dashboard interfaces for the patient and doctor, and fully implemented features for messaging, prescription management and appointment management.

During the implementation phase, my choice of using IntelliJ as the IDE tool turned out to be very beneficial, as project management, error debugging and GitLab integration were all seamless due to its unique tools.

A problem I encountered however, was choosing which coding methods to use to actually implement my features. Due to the complexity of some core functionalities, the ways in which they could be coded varied massively, and deciding the coding logic for each feature was a problem, but not one that turned out to be that big of an issue.

A different approach I would consider in hindsight, is to use **Maven** as the build tool for my Spring Boot project, as I feel it is less complex and more predictable as compared to Gradle.

Implement role-based access control while ensuring data protection, validation and security

My system accurately implements role-based access control for patients and doctors, ensuring robust user authentication for access to sensitive data and role-based features. Spring Security, BCrypt and MySQL combine to create a structured security framework, that ensures data is transmitted securely, validated before storing, and is encrypted and protected in databases.

A potential security feature to implement in hindsight is Cross-Site Request Forgery protection, as this would further prevent the risk of unauthenticated user attacks towards the system.

Create a functional database system using MySQL

MySQL is effectively used in my system to create a secure relational database, and uphold the data integrity of all entities in the system.

My approach of using MySQL to handle the data model was very helpful, as I was able to modify any incorrectly entered data easily using MySQL code, and delete null records from previous testing sessions. I would not change anything about this approach.

Develop key system features being appointment scheduling, prescription management, and secure messaging

I successfully designed and implemented key, functional and crucial features for the chronic care healthcare management process, being appointment management, prescription management, and a secure communication system. I expanded on this during development, by also adding features I deemed to be helpful to both patients and doctors, such as the health assistant and patient statistics features.

An area for improvement in my opinion, would be to develop the prescription and messaging systems with the same format as the appointment system. That is by making a single designated page for each type of user, that is very JavaScript heavy and cohesive, rather than using multiple pages with form and list elements to create the

feature. I feel the appointment system approach is more organised, has a better user interface, and produces a better user experience.

Implement and undergo various tests for the system

I was able to efficiently create a rigorous testing framework for my system, comprising of various testing types, focusing on both the functionality of the system and testing end-user compatibility.

An alternative approach I considered for the system testing phase, was also using a series of JUnit tests, that focused on user interface functionality and intended on using browser automation to navigate the system and input test data. Not only is this method tedious, but it's rather unnecessary due to the fact that it could simply be tested more efficiently through a series of human black box tests and end-to-end tests with the same functions, which is what I eventually did.

During the user testing phase, the feedback I received gave me important insight into the system from the target audience perspective, and gave me more ideas and useful features that could have been implemented into the system. With hindsight in mind, an alternative approach I would have taken would be to cooperate with end users from the very beginning, allowing me to have implemented such features from early on.

Learn to account for the issues of privacy and security

During the course of the project, I underwent extensive research into the importance of security in software development and software systems. This research more importantly extended into the privacy and security importances in digital healthcare solutions, allowing me to discover multiple laws and regulations regarding data protection and security, such as GDPR and HIPAA.

My knowledge acquisition then extended into the actual security design process of my software system. I realised the significance of data protection and security when configuring Spring Security and BCrypt methods for my login and registration processes. I learnt that a system that fails to implement these methods to encrypt data, offer role-based access control, authorise users, securely store data, and transmit data securely, poses an extremely high risk to data protection, causing potential data breaches, privacy and security concerns, financial losses, reputation damage and legal liabilities.

To summarise, through the course of the project I successfully achieved my aims and objectives that were originally set out for the project. The system produced is an accurate reflection of my functional requirements, and successfully fixes the problem of poor chronic health management in healthcare systems.

6.1.2 System Limitations

Following further critical analysis, my system still has some limitations that leave minor areas for improvement:

- No administrative role for system management
- No current integration with existing electronic health platforms
- No feature for patients to input updated health data or symptoms

These limitations can be accepted due to the extreme complexity of the implementation for the features, but still identify key areas my system could improve upon in the future.

6.2 Real World Impact

My system has a significant potential impact on people, businesses and the society as a whole, with numerous benefits. Firstly for people, my system offers improved healthcare management for chronically ill patients, allowing remote access to crucial services such as prescription and appointment management. The ability to also view and track their health trends and be actively involved in decision making, will massively impact the quality of their healthcare treatment.

For businesses, this system provides healthcare providers with a platform for efficient patient management and monitoring, with optimal features to increase business workflow, such as prescription refill requests and appointment scheduling. These features positively impact businesses, by decreasing work burden on staff, maximising time, and reducing costs.

However, the system handles highly sensitive health data that is vulnerable to data breaches, posing risks and highlighting privacy and security concerns. There is also risk of over reliance on technology, as in-person healthcare is still necessary. The digital divide is always a risk with technology applications, as in this scenario, some patients may not have equal access to technology, or the skills required to use a web application.

Many technologies used in the course of the project such as database management, and role-based access control, are widely used today in common enterprise applications, making the product marketable to hospitals, clinics and pharmacies, with the huge potential of becoming a commercial success.

6.3 Personal Development

Before starting my project, I had no previous knowledge of the impact and power of technology in healthcare. This project gave me the opportunity to learn about this, and the problems the average chronically ill patient faces while managing their healthcare and conditions.

I had minimal experience in the Spring Boot framework before this project. Months of coding, multiple errors faced, and hours of further learning research, allowed me to acquire the skills necessary towards designing and building a high quality, robust, reliable and secure web application in Spring Boot. I now understand the fundamentals of data encryption in applications, and have gained further skills in JavaScript. I also acquired knowledge needed to design and create an interactive user interface, learning coding methods to grab user attention, ease user navigation, and improve user satisfaction.

As someone who aims to go into the field of web development in the future, this project provided the perfect opportunity to gain experience and the necessary skills required for the field.

7. Conclusion

To conclude, I have developed a fully functional web-based application to manage chronic patients health, while successfully meeting all the aims and objectives of the project that were set out in the beginning. For this reason, I would call this project a success.

I am pleased with the outcome of the system and the project, but there are still areas for improvement, in regards to the limitations stated earlier.

Word count without title page, declaration and references: **10,887**

8. Bibliography

[1] J. Wagner, “Chronic Disease Management: Improving Outcomes, Reducing Costs | Crown Family School of Social Work, Policy, and Practice,” *crownschool.uchicago.edu*. https://crownschool.uchicago.edu/student-life/advocates-forum/chronic-disease-management-improving-outcomes-reducing-costs (accessed Apr. 05, 2025).

[2] N. Samrin, “Spring Boot Framework: Features, Benefits, and Use Cases - StaticMania,” *StaticMania*, Dec. 11, 2024. https://staticmania.com/blog/spring-boot (accessed Apr. 04, 2025).

[3] A. James, “How To Safely and Securely Handle Sensitive Information at Work,” *Hut Six*. https://www.hutsix.io/how-to-handle-sensitive-information-at-work/ (accessed Apr. 05, 2025).

[4] The Health Foundation, “2.5 Million More People in England Projected to Be Living with Major Illness by 2040,” *The Health Foundation*, Jul. 25, 2023. https://www.health.org.uk/press-office/press-releases/25-million-more-people-in-england-projected-to-be-living-with-major (accessed Apr. 05, 2025).

[5] M. Duda-Sikuła and D. Kurpas, “Enhancing Chronic Disease Management: Personalized Medicine Insights from Rural and Urban General Practitioner Practices,” *Journal of Personalized Medicine*, vol. 14, no. 7, p. 706, Jul. 2024, doi: https://doi.org/10.3390/jpm14070706.

[6] S. Alder, “Effects of poor communication in healthcare,” *The HIPAA Journal*, Jan. 02, 2025. https://www.hipaajournal.com/effects-of-poor-communication-in-healthcare/ (accessed Apr. 05, 2025).

[7] “Digital Health | History Timeline,” *History Timelines*, 2019. https://historytimelines.co/timeline/digital-health (accessed Apr. 05, 2025).

[8] E. P. Ambinder, “A History of the Shift Toward Full Computerization of Medicine,” *Journal of Oncology Practice*, vol. 1, no. 2, pp. 54–56, Jul. 2005, doi: https://doi.org/10.1200/jop.2005.1.2.54.

[9] A. Grudniewicz, C. S. Gray, P. Boeckxstaens, J. D. Maeseneer, and J. W. Mold, “Operationalizing the Chronic Care Model with Goal-Oriented Care,” *The Patient: Patient-Centered Outcomes Research*, vol. 16, no. 6, pp. 569–578, 2023, doi: https://doi.org/10.1007/s40271-023-00645-8.

[10] H. R. Holman, “The Relation of the Chronic Disease Epidemic to the Health Care Crisis,” *ACR Open Rheumatology*, vol. 2, no. 3, pp. 167–173, Feb. 2020, doi: https://doi.org/10.1002/acr2.11114.

[11] J. C. Moses, S. Adibi, S. M. Shariful Islam, N. Wickramasinghe, and L. Nguyen, “Application of Smartphone Technologies in Disease Monitoring: A Systematic Review,” *Healthcare*, vol. 9, no. 7, p. 889, Jul. 2021, doi: https://doi.org/10.3390/healthcare9070889.

[12] Ma, Y., Zhao, C., Zhao, Y. "Telemedicine application in patients with chronic disease: a systematic review and meta-analysis," *BMC Medical Informatics and Decision Making*, vol. 22, no. 233, 2022. Available: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-022-01845-2.

[13] A.A. Seixas, I.M. Olaye, S.P. Wall, P. Dunn "Optimizing Healthcare Through Digital Health and Wellness Solutions to Meet the Needs of Patients With Chronic Disease During the COVID-19 Era," *Frontiers in Public Health*, vol. 9, 2021. Available: https://www.frontiersin.org/articles/10.3389/fpubh.2021.667654/full.

[14] M. L. Taylor *et al.*, “Digital health experiences reported in chronic disease management: An umbrella review of qualitative studies,” *Journal of Telemedicine and Telecare*, vol. 28, no. 10, pp. 705–717, Nov. 2022, doi: https://doi.org/10.1177/1357633x221119620.

[15] J. Doyle *et al.*, “A Digital Platform to Support Self-management of Multiple Chronic Conditions (ProACT): Findings in Relation to Engagement During a One-Year Proof-of-Concept Trial,” *Journal of Medical Internet Research*, vol. 23, no. 12, p. e22672, Dec. 2021, doi: https://doi.org/10.2196/22672.

[16] A. S. Young *et al.*, “Information Technology to Support Improved Care For Chronic Illness,” *Journal of General Internal Medicine*, vol. 22, no. 3, pp. 425–430, Dec. 2007, doi: https://doi.org/10.1007/s11606-007-0303-4.

[17] C. Pong, R. Marjorie, Y. C. Tham, and E. Lum, “Current Implementation of Digital Health in Chronic Disease Management: Scoping Review,” *Journal of Medical Internet Research*, vol. 26, pp. e53576–e53576, Dec. 2024, doi: https://doi.org/10.2196/53576.

[18] W. N. Price and I. G. Cohen, “Privacy in the Age of Medical Big Data,” *Nature Medicine*, vol. 25, no. 1, pp. 37–43, 2019, doi: https://doi.org/10.1038/s41591-018-0272-7.

[19] netalit, “What Is the HIPAA Security Rule?,” *Check Point Software*, Feb. 18, 2025. https://www.checkpoint.com/cyber-hub/cyber-security/what-is-hipaa-compliance/what-is-the-hipaa-security-rule/ (accessed Apr. 06, 2025).

[20] “What Is HIPAA Compliance? HIPAA Laws & Rules | Proofpoint UK,” *Proofpoint*, May 31, 2021. https://www.proofpoint.com/uk/threat-reference/hipaa-compliance (accessed Apr. 06, 2025).

[21] L. Galloway, “HIPAA security rule: Compliance for beginners | Pabau CRM,” *Pabau*, Nov. 17, 2023. https://pabau.com/blog/hipaa-security-rule-requirements/ (accessed Apr. 06, 2025).

[22] HIPAA Journal, “HIPAA encryption requirements,” *HIPAA Journal*, 2023. https://www.hipaajournal.com/hipaa-encryption-requirements/ (accessed Apr. 06, 2025).

[23] “NHS App hits over 30 million sign-ups,” *GOV.UK*, Dec. 31, 2022. https://www.gov.uk/government/news/nhs-app-hits-over-30-million-sign-ups (accessed Apr. 07, 2025).

[24] S. E. Ross and C.-T. Lin, “The Effects of Promoting Patient Access to Medical Records: A Review,” *Journal of the American Medical Informatics Association*, vol. 10, no. 2, pp. 129–138, Mar. 2003, doi: https://doi.org/10.1197/jamia.m1147.

[25] T. Pathe, “Paymentus Enhances Healthcare Billing and Payments Integration With Epic MyChart,” *The Fintech Times*, Oct. 06, 2022. https://thefintechtimes.com/paymentus-enhances-healthcare-billing-and-payments-integration-with-epic-mychart/ (accessed Apr. 07, 2025).

[26] rishabh, “Epic EHR Explained: Features, Benefits, and Impact on Healthcare,” *HealthConnect*, Feb. 12, 2025. https://healthconnect.mindbowser.com/epic-ehr-explained/ (accessed Apr. 07, 2025).

[27] I. QliqSOFT, “Epic Patient Portal Advantages and Limitations,” *Qliqsoft.com*, Jun. 26, 2024. https://www.qliqsoft.com/research/epic-patient-portal-advantages-and-limitations (accessed Apr. 07, 2025).

[28] World Health Organisation, “The critical role of accessibility in health information,” *www.who.int*, Apr. 25, 2023. https://www.who.int/europe/news-room/feature-stories/item/the-critical-role-of-accessibility-in-health-information (accessed Apr. 07, 2025).

[29] M. Stone, “Role Based Access Control Explained: RBAC Security Overview,” *Concentric AI*, Jan. 07, 2025. https://concentric.ai/how-role-based-access-control-rbac-helps-data-security-governance/ (accessed Apr. 07, 2025).

[30] D. Helminski *et al.*, "Dashboards in Health Care Settings: Protocol for a Scoping Review," *JMIR Research Protocols*, vol. 11, no. 3, p. e34894, Mar. 2022, doi: https://doi.org/10.2196/34894.

[31] A. E. McGuffee, K. Chillag, A. Johnson, R. Richardson, H. Williams, and J. Hartos, "Effects of Routine Checkups and Chronic Conditions on Middle-Aged Patients with Diabetes," *Advances in Preventive Medicine*, vol. 2020, pp. 1–8, Feb. 2020, doi: https://doi.org/10.1155/2020/4043959.

[32] FasterCapital, "Feedback loops: Feedback Loops in Healthcare: Improving Patient Care with Feedback Loops - FasterCapital," *FasterCapital*, 2024. https://fastercapital.com/content/Feedback-loops--Feedback-Loops-in-Healthcare--Improving-Patient-Care-with-Feedback-Loops.html (accessed Apr. 07, 2025).

[33] M. Hägglund, B. McMillan, R. Whittaker, and C. Blease, "Patient empowerment through online access to health records," *BMJ*, vol. 378, no. e071531, Sep. 2022, doi: https://doi.org/10.1136/bmj-2022-071531.

[34] P. Rome, "What are Non Functional Requirements — With Examples," *Perforce Software*, Aug. 11, 2020. https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples (accessed Apr. 14, 2025).

[35] O. Phelps, "Guide to Non-Functional Requirements: Types and Examples | Insight | Box UK," *www.boxuk.com*, Nov. 13, 2020. https://www.boxuk.com/insight/guide-to-non-functional-requirements-types-and-examples/ (accessed Apr. 15, 2025).

[36] A. Donovan, "What Is a Performance Management System and Why Are They Important?," *Donovan Training Associates*, Jan. 30, 2024. https://www.donovantrainingassociates.co.uk/blog/performance-management-systems (accessed Apr. 15, 2025).

[37] K. R. Dufendach, A. Navarro-Sainz, and K. L. Webster, "Usability of human-computer interaction in neonatal care," *Seminars in Fetal and Neonatal Medicine*, vol. 27, no. 4, p. 101395, Oct. 2022, doi: https://doi.org/10.1016/j.siny.2022.101395.

[38] Concepta, "The Importance of Scalability In Software DesignSoftware Engineering," *www.conceptatech.com*, Aug. 20, 2020. https://www.conceptatech.com/blog/importance-of-scalability-in-software-design (accessed Apr. 15, 2025).

[39] N. S. Sahu, "Understanding Thymeleaf: A Powerful Java Template Engine," *DEV Community*, May 14, 2024. https://dev.to/nikhilxd/understanding-thymeleaf-a-powerful-java-template-engine-3bnh (accessed Apr. 08, 2025).

[40] A. Patel, “Integrating Spring Boot with MySQL/PostgreSQL: A Comprehensive Guide,” *Medium*, Sep. 24, 2024. https://medium.com/@FullStackSoftwareDeveloper/integrating-spring-boot-with-mysql-postgresql-a-comprehensive-guide-725696e22dcd (accessed Apr. 08, 2025).

[41] “Spring Projects,” *Spring.io*, 2019. https://spring.io/projects/spring-security (accessed Apr. 08, 2025).

[42] Roshan Farakate, “Understanding DTOs in Spring Boot: A Comprehensive Guide,” *Medium*, Jul. 22, 2024. https://medium.com/@roshanfarakate/understanding-dtos-in-spring-boot-a-comprehensive-guide-20e2b8101ee6 (accessed Apr. 11, 2025).

[43] P. Kumar, “Mastering JPA Inheritance Strategies: Hibernate 6.x JPA 3.x Spring Boot 3.x,” *Medium*, Mar. 24, 2024. https://medium.com/@iampraveenkumar/mastering-jpa-inheritance-strategies-hibernate-6-x-jpa-3-x-spring-boot-3-x-06eecac1147a (accessed Apr. 11, 2025).

[44] A. Ot, “Relational Data Model 101: Key Components & Benefits” *Datamation*, Nov. 29, 2023. https://www.datamation.com/big-data/relational-data-model/ (accessed Apr. 11, 2025).

[45] Brendan, “Understanding CSS: Advantages and Disadvantages of Inline, Internal, and External Styles” *DEV Community*, Jun. 22, 2024. https://dev.to/brendan_frasser/understanding-css-advantages-and-disadvantages-of-inline-internal-and-external-styles-glk (accessed Apr. 10, 2025).

[46] V. Paget, “Features vs Functionality: How to accurately compare software systems | Orah Blog” *www.orah.com*, Jun. 18, 2017. https://www.orah.com/blog/features-vs-functionality (accessed Apr. 11, 2025).

[47] A. Hughes, “Encryption vs. Hashing vs. Salting - What’s the Difference?” *www.pingidentity.com*, Mar. 01, 2022. https://www.pingidentity.com/en/resources/blog/post/encryption-vs-hashing-vs-salting.html (accessed Apr. 12, 2025).

[48] M. White, “How tough is bcrypt to crack? And can it keep passwords safe?” *Specops Software*, Nov. 15, 2023. https://specopssoft.com/blog/hashing-algorithm-cracking-bcrypt-passwords/ (accessed Apr. 12, 2025).

[49] Simplilearn, “What is Intellij IDEA - A Comprehensive Guide,” *Simplilearn.com*, Oct. 23, 2023. https://www.simplilearn.com/what-is-intellij-idea-article (accessed Apr. 14, 2025).

[50] M. Yi, “Data Viz Color Selection Guide | Atlassian,” *Atlassian*, 2024. https://www.atlassian.com/data/charts/how-to-choose-colors-data-visualization (accessed Apr. 14, 2025).

[51] Instandart, “Interactive Data Dashboards: Trends and Best Practices - Instandart,” *Instandart* -, Apr. 07, 2025. https://instandart.com/blog/bespoke-software-development/interactive-data-dashboards-trends-and-best-practices/ (accessed Apr. 15, 2025).

[52] Welkin, “3 Reasons Why You Need Patient-Centered Care,” *Welkin Health*, 2020. https://welkinhealth.com/patient-centered-care/ (accessed Apr. 16, 2025).

[53] M. Clark and S. Bailey, *Chatbots in Health Care: Connecting Patients to Information: Emerging Health Technologies*. Ottawa (ON): Canadian Agency for Drugs and Technologies in Health, 2024. Accessed: Apr. 16, 2025. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK602381/

[54] J. McGuire, “Why Is Software Testing Important in the Software Development Lifecycle?,” *Pulsion Technology*, May 06, 2024. https://www.pulsion.co.uk/blog/why-is-software-testing-important-in-the-software-development-lifecycle/ (accessed Apr. 18, 2025).

[55] K. Yasar, “What is Software Testing? Definition, Types and Importance,” *TechTarget*, Aug. 2022. https://www.techtarget.com/whatis/definition/software-testing (accessed Apr. 18, 2025).

[56] R. Miecznik, “The Importance And Benefits of Unit Testing,” *CodiLime*, Jun. 22, 2023. https://codilime.com/blog/unit-testing/ (accessed Apr. 18, 2025).

[57] T. Shah, “Unit Test Writing with AI vs. Manual Methods: A Deep Dive,” *Zencoder.ai*, Sep. 18, 2024. https://zencoder.ai/blog/ai-vs-manual-unit-testing (accessed Apr. 18, 2025).

[58] T. Collins, “Different Types of Testing in Software | BrowserStack,” *BrowserStack*, Aug. 22, 2024. http://browserstack.com/guide/types-of-testing (accessed Apr. 20, 2025).

[59] J. Anderson, “End-User Testing: What It Is, Why It’s Important, and How to Perform It | Loop11,” *Loop11 | Easiest Online Usability Testing Tool*, Jul. 16, 2024. https://www.loop11.com/end-user-testing-what-it-is-why-its-important-and-how-to-perform-it/ (accessed Apr. 20, 2025).