**Medi-Log: The Smart Health Hub**

****

**Muhammad Talha Shafiq (BSE213130)**

**Anoosha Javed (BSE213124)**

**Supervised by**

**Mr. Wasif Ali**

**Fall 2026**

**BS Software Engineering**

**Department of Software Engineering**

**Capital University of Science & Technology, Islamabad**

**Table of Content**

[**Chapter 1** 8](#_Toc215319015)

[**1.** **Introduction** 8](#_Toc215319016)

[**1.1.** **Project Introduction** 8](#_Toc215319017)

[**1.2.** **Objectives** 8](#_Toc215319018)

[**1.3.** **Existing Examples / Solutions** 9](#_Toc215319019)

[**1.4.** **Our Proposed System** 10](#_Toc215319020)

[**1.4.1.** **Secure Patient Data Management** 10](#_Toc215319021)

[**1.4.2.** **Advanced Appointment Scheduling, Reminders & Management** 10](#_Toc215319022)

[**1.4.3.** **Digital Prescription Delivery** 10](#_Toc215319023)

[**1.4.4.** **Patient Report Management** 10](#_Toc215319024)

[**1.4.5.** **Comprehensive History Maintenance** 11](#_Toc215319025)

[**1.4.6.** **Real-Time Communication Module** 11](#_Toc215319026)

[**1.4.7.** **Data Encryption & Privacy** 11](#_Toc215319027)

[**1.4.8.** **Cross-Device Accessibility** 11](#_Toc215319028)

[**1.5.** **Business Scope** 11](#_Toc215319029)

[**1.6.** **Tools and Technologies** 12](#_Toc215319030)

[**1.6.1.** **PostgreSQL** 12](#_Toc215319031)

[**1.6.2.** **Python** 12](#_Toc215319032)

[**1.6.3.** **Streamlit** 13](#_Toc215319033)

[**1.7.** **Project Timeline** 14](#_Toc215319034)

[**1.8.** **Project Work Breakdown** 15](#_Toc215319035)

[**Chapter 2** 16](#_Toc215319036)

[**2.** **Requirement Specification and Analysis** 16](#_Toc215319037)

[**2.1.** **Functional Requirements** 16](#_Toc215319038)

[**2.2.** **Non-Functional Requirements** 17](#_Toc215319039)

[**2.3.** **System Use Case Modeling** 18](#_Toc215319040)

[**2.3.1.** **Use Case 1: Register/Login** 19](#_Toc215319041)

[**2.3.2.** **Use Case 2: Upload Medical Record** 21](#_Toc215319042)

[**2.3.3.** **Use Case 3: Book Appointment** 22](#_Toc215319043)

[**2.3.4.** **Use Case 4: Issue E-Prescription** 23](#_Toc215319044)

[**2.3.5.** **Use Case 5: Communicate with Doctor** 24](#_Toc215319045)

[**2.3.6.** **Use Case 6: Forgot Password** 25](#_Toc215319046)

[**2.3.7.** **Use Case 7: Access Cross-Device Data** 26](#_Toc215319047)

[**2.3.8.** **Use Case 8: View/Download E-Prescription** 27](#_Toc215319048)

[**2.3.9.** **Use Case 9: View Medical History** 28](#_Toc215319049)

[**2.3.10.** **Use Case 10: Review Health Insights** 29](#_Toc215319050)

[**2.3.11.** **Use Case 11: Update Medical History** 30](#_Toc215319051)

[**2.3.12.** **Use Case 12: Audit Logs for Prescriptions** 31](#_Toc215319052)

[**2.3.13.** **Use Case 13: Role Based Access Control** 32](#_Toc215319053)

[**2.3.14.** **Use Case 14: Receive Notification** 33](#_Toc215319054)

[**2.4.** **System Sequence Diagram** 33](#_Toc215319055)

[**2.5.** **Domain Model** 39](#_Toc215319056)

[**2.6.** **User Interface Design (Prototype)** 40](#_Toc215319057)

[**Chapter 3** 52](#_Toc215319058)

[**3.** **System Design** 52](#_Toc215319059)

[**3.1.** **Software Architecture Diagram** 52](#_Toc215319060)

[**3.2.** **Class Diagram** 53](#_Toc215319061)

[**3.3.** **Sequence Diagram** 53](#_Toc215319062)

[**3.4.** **Entity Relationship Diagram** 54](#_Toc215319063)

[**3.5.** **Database Schema** 54](#_Toc215319064)

[**Chapter 4** 55](#_Toc215319065)

[**4.** **Software Description** 55](#_Toc215319066)

[**4.1.** **Coding Standards** 55](#_Toc215319067)

[**4.2.** **Development Environment** 55](#_Toc215319068)

[**4.3.** **Software Description** 57](#_Toc215319069)

[**4.3.1.** **Registration** 57](#_Toc215319070)

[**4.3.2.** **Login** 58](#_Toc215319071)

[**4.3.3.** **Patient Dashboard** 59](#_Toc215319072)

[**4.3.4.** **Doctor Dashboard** 63](#_Toc215319073)

[**Chapter 5** 69](#_Toc215319074)

[**5.** **Software Testing** 69](#_Toc215319075)

[**5.1.** **Testing Methodology** 69](#_Toc215319076)

[**5.2.** **Testing Environment** 69](#_Toc215319077)

[**5.3.** **Test Cases** 70](#_Toc215319078)

[**5.3.1.** **Test Case 1: User Login with Valid Credentials** 70](#_Toc215319079)

[**5.3.2.** **Test Case 2: User Login with Invalid Credentials** 71](#_Toc215319080)

[**5.3.3.** **Test Case 3: Patient Registration** 72](#_Toc215319081)

[**5.3.4.** **Test Case 4: Appointment Booking** 73](#_Toc215319082)

[**5.3.5.** **Test Case 5: View Medical Records** 74](#_Toc215319083)

[**5.3.6.** **Test Case 6: View Prescription** 75](#_Toc215319084)

[**5.3.7.** **Test Case 7: View Profile (Patient)** 76](#_Toc215319085)

[**5.3.8.** **Test Case 8: Chat with Doctor** 77](#_Toc215319086)

[**5.3.9.** **Test Case 9: View Dashboard (Patient)** 79](#_Toc215319087)

[**5.3.10.** **Test Case 10: View Dashboard (Doctor)** 80](#_Toc215319088)

[**5.3.11.** **Test Case 11: View Schedule** 82](#_Toc215319089)

[**5.3.12.** **Test Case 12: Manage Appointments (Doctor)** 84](#_Toc215319090)

[**5.3.13.** **Test Case 13: Manage Treatment** 87](#_Toc215319091)

[**5.3.14.** **Test Case 14: View Prescription (Doctor)** 89](#_Toc215319092)

[**5.3.15.** **Test Case 15: View Profile (Doctor)** 91](#_Toc215319093)

[**5.3.16.** **Test Case 16: Chat with Patient** 93](#_Toc215319094)

**List of Tables**

[Table 1 Existing Solutions 9](#_Toc215319095)

[Table 2 Functional Requirements 16](#_Toc215319096)

[Table 3 Non-functional Requirements 17](#_Toc215319097)

[Table 4 Register/Login 19](#_Toc215319098)

[Table 5 Upload Medical Record 21](#_Toc215319099)

[Table 6 Book Appointment 22](#_Toc215319100)

[Table 7 Issue E-Prescription 23](#_Toc215319101)

[Table 8 Communicate with Doctor 24](#_Toc215319102)

[Table 9 Forgot Password 25](#_Toc215319103)

[Table 10 Access Cross Device Data 26](#_Toc215319104)

[Table 11 View/Download E-Prescription 27](#_Toc215319105)

[Table 12 View Medical History 28](#_Toc215319106)

[Table 13 Review Health Insights 29](#_Toc215319107)

[Table 14 Update Medical Records 30](#_Toc215319108)

[Table 15 Audit Logs for Prescriptions 31](#_Toc215319109)

[Table 16 Role-Based Access Control 32](#_Toc215319110)

[Table 17 Receive Notifications 33](#_Toc215319111)

[Table 18 User Login with Valid Credentials 70](#_Toc215319112)

[Table 19 User Login with Invalid Credentials 71](#_Toc215319113)

[Table 20 Patient Registration 72](#_Toc215319114)

[Table 21 Appointment Booking 73](#_Toc215319115)

[Table 22 View Medical Record 74](#_Toc215319116)

[Table 23 View Prescriptions 75](#_Toc215319117)

[Table 24 View Profile (Patient) 76](#_Toc215319118)

[Table 25 Chat with Doctor 78](#_Toc215319119)

[Table 26 View Dashboard (Patient) 80](#_Toc215319120)

[Table 27 View Doctor Dashboard 81](#_Toc215319121)

[Table 28 Manage Schedule 83](#_Toc215319122)

[Table 29 View Appointments 86](#_Toc215319123)

[Table 30 Manage Treatment 89](#_Toc215319124)

[Table 31 View Prescription 90](#_Toc215319125)

[Table 32 View Profile (Doctor) 92](#_Toc215319126)

[Table 33 Chat with Patient 94](#_Toc215319127)

**List of Figures**

[Figure 1 PosgreSQL Logo 12](#_Toc215318971)

[Figure 2 Python Logo 13](#_Toc215318972)

[Figure 3 Streamlit Logo 13](#_Toc215318973)

[Figure 4 Project Timeline 14](#_Toc215318974)

[Figure 5 Project Work Breakdown 15](#_Toc215318975)

[Figure 6 Patient Use Case Diagram 18](#_Toc215318976)

[Figure 7 Doctor Use Case Diagram 19](#_Toc215318977)

[Figure 8 User Login/ Register 33](#_Toc215318978)

[Figure 9 Upload Medical Reports 34](#_Toc215318979)

[Figure 10 Book Appointment 34](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318980)

[Figure 11 Reschedule/Cancel Appointments 34](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318981)

[Figure 12 Share Reports 35](#_Toc215318982)

[Figure 13 Communicate with Doctor 35](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318983)

[Figure 14 Access Cross Device Data 36](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318984)

[Figure 15 View/Download E-Prescription 37](#_Toc215318985)

[Figure 16 View/Download Medical History 37](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318986)

[Figure 17 View Health Insights 37](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318987)

[Figure 18 Update Medical History 38](#_Toc215318988)

[Figure 19 Issue E-Prescriptions 38](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318989)

[Figure 20 Audit Logs for Prescriptions 38](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318990)

[Figure 21 Receive Notifications 39](#_Toc215318991)

[Figure 22 Domain Model 39](#_Toc215318992)

[Figure 23 Home Screen 40](#_Toc215318993)

[Figure 24 Login Screen 41](#_Toc215318994)

[Figure 25 Registration Screen 41](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318995)

[Figure 26 Feature Screen 42](#_Toc215318996)

[Figure 27 About Screen 43](#_Toc215318997)

[Figure 28 Contact Us Screen 43](file:///C:\Users\ALRASHIDS\Desktop\projects\NewSmartHealthHub\document.docx#_Toc215318998)

[Figure 29 Patient Dashboard Screen 45](#_Toc215318999)

[Figure 30 Find Doctor Screen 45](#_Toc215319000)

[Figure 31 Schedule/Cancel Appointment Screen 47](#_Toc215319001)

[Figure 32 Labs Screen 47](#_Toc215319002)

[Figure 33 Medical History Screen 48](#_Toc215319003)

[Figure 34 Make Appointment Screen 48](#_Toc215319004)

[Figure 35 Payment Screen 49](#_Toc215319005)

[Figure 36 Doctor Dashboard Screen 49](#_Toc215319006)

[Figure 37 Patient Info Screen 50](#_Toc215319007)

[Figure 38 Appointment Screen 50](#_Toc215319008)

[Figure 39 Settings Screen 51](#_Toc215319009)

[Figure 40 Software Architecture Diagram 52](#_Toc215319010)

[Figure 41 Class Diagram 53](#_Toc215319011)

[Figure 42 Sequence Diagram 53](#_Toc215319012)

[Figure 43 Entity Relationship Diagram 54](#_Toc215319013)

[Figure 44 Database Schema 54](#_Toc215319014)

# **Chapter 1**

# **Introduction**

# **Project Introduction**

Healthcare is evolving rapidly, and digital solutions are at the forefront of this transformation. Traditional paper-based record-keeping systems and fragmented digital solutions have paved the way for comprehensive platforms that integrate appointment management, digital prescriptions, robust communication, and extensive history maintenance. This proposal outlines an advanced Digital Health Record Mobile Application that incorporates key features of leading platforms such as Oladoc, Sehat Zindagi, and Marham, while further enhancing them to meet modern patient and provider needs.

# **Objectives**

This enhanced application aims to:

* **Digital Transformation:** Transition from paper-based records to a secure, cloud-based digital system.
* **Streamline appointment management:** Offer a sophisticated scheduling system with automated reminders, rescheduling, and cancellation options.
* **Facilitate digital prescriptions:** Provide a secure and paperless prescription process accessible by patients and healthcare providers.
* **Empower patient self-management:** Enable patients to update and maintain their medical history and reports in real time.
* **Enhance real-time communication:** Introduce a dedicated communication module for secure chats, audio, and video consultations.
* **Enhance Decision-Making:** Transform health data into actionable insights with predictive analytics and dashboards
* **Ensure cross-device accessibility and data privacy:** Maintain real-time synchronization across devices using robust encryption and access control measures.

# **Existing Examples / Solutions**

The following table demonstrates the comparison of our proposed system with the existing systems.

Table 1 Existing Solutions

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **Our Proposed System** | **Microsoft HealthVault (Discontinued)** | **Apple**  **Health App** | **MyChart by Epic** | **Google Fit** | **Oladoc** | **Sehat Zindagi** | **Marham** |
| **Centralized**  **Digital Record Storage** | Yes | Yes | No | No | Yes | Limited | Limited | Limited |
| **Health Tracking**  **(Vitals, Steps, etc.)** | Yes | No | Basic | No | Yes | No | No | No |
| **Hospital & Clinic Integration** | Yes | Limited | No | Yes | No | Yes | Yes | Yes |
| **Health Insights Through**  **Wearable Data** | Yes | No | Limited | No | Basic | No | No | No |
| **Real-Time Health Monitoring**  **(Wearable Integration)** | Yes | No | Basic | No | Basic | No | No | No |
| **Advanced Appointment**  **Scheduling** | Yes | No | No | Limited | No | Yes | Yes | Yes |
| **Digital Prescription**  **Delivery** | Yes | No | No | No | No | Yes | Yes | Yes |
| **Comprehensive**  **History Maintenance** | Yes | No | No | No | No | Yes | Yes | Yes |
| **Real-Time Communication (Chat, Voice,**  **Video)** | Yes | No | No | No | No | Yes | Yes | Yes |
| **Data Security & Privacy (Encryption,**  **Authentication)** | Yes | Limited | Limited | Yes | Limited | Limited | Limited | Limited |

# **Our Proposed System**

The proposed application is an all-inclusive digital health solution designed to address current challenges and integrate features that reflect the best practices of leading platforms. The system’s components include:

## **Secure Patient Data Management**

* **Cloud Storage:** Securely stores all medical records, lab results, and patient histories.
* **Role-Based Access:** Allows doctors and patients differentiated access levels to ensure data security.
* **End-to-End Encryption:** Maintains confidentiality and compliance with healthcare regulations.

## **Advanced Appointment Scheduling, Reminders & Management**

* + **Comprehensive Scheduling:** Patients can book, reschedule, or cancel appointments with ease using real-time doctor availability and location filters.
  + **Automated Reminders:** Sends notifications for appointments, medication timings, and follow-up visits to improve adherence.

## **Digital Prescription Delivery**

* + **E-Prescriptions:** Doctors can issue secure digital prescriptions that can be stored and shared with pharmacies, reducing the risks of errors and paper-based processes.
  + **Audit Trails:** Maintains logs for all issued prescriptions to ensure accountability and traceability.

## **Patient Report Management**

* **Document Upload & Organization:** Patients can upload, categorize, and manage their medical reports, lab results, and imaging studies in a centralized repository.
* **Easy** **Access & Sharing**: Enables secure sharing of reports with healthcare providers for informed decision-making.

## **Comprehensive History Maintenance**

* **Detailed Record Keeping:** Offers a module dedicated to maintaining a complete and up- to-date record of a patient’s medical history, including previous appointments, diagnoses, treatments, and prescriptions.
* **Self-Update Capability:** Empowers patients to actively update their medical history, ensuring that records are current and comprehensive.

## **Real-Time Communication Module**

* **Multi-Channel Interaction:** Provides secure in-app messaging, voice, and video call features for direct patient-doctor communication.
* **Integrated Consultations:** Facilitates seamless transition from appointment booking to consultation without needing to switch platforms.
* **Instant Notifications:** Alerts both patients and providers when new messages or

## **Data Encryption & Privacy**

* **Robust Security Measures:** Implements advanced encryption, secure authentication, and regular security audits to safeguard sensitive data.
* **Compliance:** Adheres to HIPAA and other relevant data protection regulations.

## **Cross-Device Accessibility**

* **Seamless Experience:** Ensures real-time synchronization of medical records across smartphones, tablets, and desktops, enabling healthcare providers to access updated patient information anytime, anywhere.

# **Business Scope**

The Digital Health Record Mobile Application is aimed at serving hospitals, clinics, pharmacies, and individual patients by offering a secure and efficient medical record management system.

* **Revenue Model:** Subscription-based services for institutional clients, with basic free access for individual users and optional premium features such as AI-driven health insights.
* **Partnerships:** Collaboration with healthcare providers and institutions to promote widespread adoption and seamless integration into existing healthcare infrastructures

# **Tools and Technologies**

## **PostgreSQL**

PostgreSQL is a powerful, open-source object-relational database system known for its robustness, extensibility, and standards compliance. With decades of active development, PostgreSQL has earned a reputation for reliability, strong performance, and advanced feature support, making it a preferred choice for complex, data-intensive applications used by organizations such as Instagram, Spotify, Apple, and many others.

The PostgreSQL community continually enhances the platform, introducing new capabilities to support modern data workloads across web, cloud, mobile, and enterprise environments.

As an object-relational database, PostgreSQL will store user data, notes, and other backend information while providing advanced querying, scalability options, and strong security features, ensuring efficient and secure data management for the application.



Figure 1 PosgreSQL Logo

## **Python**

Python is a general-purpose language, which means it's designed to be used in a range of applications, including data science, software and web development, automation, and generally getting stuff done.

Python will be used to integrate data science tasks, handle back-end logic.



Figure 2 Python Logo

## **Streamlit**

Streamlit is an open-source Python framework designed for building interactive web applications quickly and easily. It allows developers to create rich, data-driven interfaces using simple Python scripts, eliminating the need for front-end development. Streamlit offers a wide range of built-in components such as widgets, charts, and layout tools, enabling rapid prototyping and visualization. It integrates smoothly with popular data libraries like Pandas, NumPy, and scikit-learn, making it ideal for analytical and machine learning applications. Streamlit also supports connections to databases like PostgreSQL, MySQL, and SQLite, ensuring flexible data management. With its secure execution model and real-time updates, it provides a responsive and user-friendly experience. Overall, Streamlit is a strong choice for building lightweight dashboards and interactive tools, including those used in medical or research settings.



Figure 3 Streamlit Logo

# **Project Timeline**

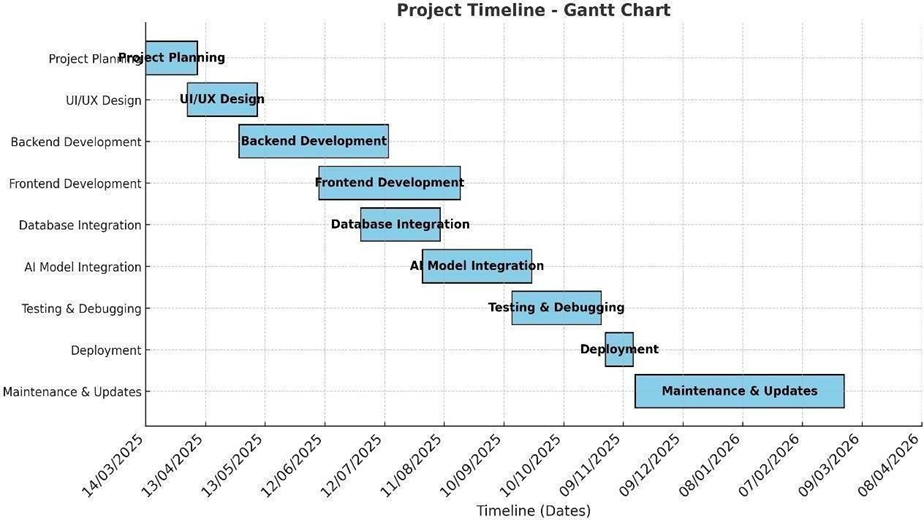


Figure 4 Project Timeline

# **Project Work Breakdown**

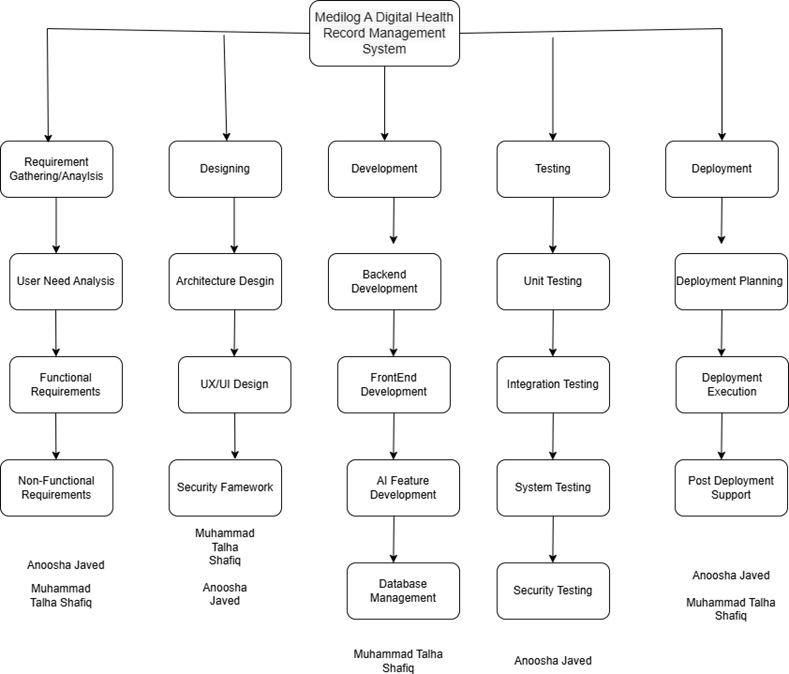


Figure 5 Project Work Breakdown

# **Chapter 2**

# **Requirement Specification and Analysis**

Requirement analysis involves gathering, examining, and refining the necessary functionalities and constraints for the proposed Medi-Log: The Smart Health Hub. This process ensures that the final system meets user expectations and aligns with real-world healthcare needs.

Stakeholders include patients, doctors, and supporting services such as cloud storage providers and authentication mechanisms.

# **Functional Requirements**

Table 2 Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Functional Requirements** | **Type** | **Status** |
| 1 | The user shall be able to register and log in to the  application. | Core | Planned |
| 2 | The system shall store and manage patient medical records  and history. | Core | Planned |
| 3 | The system shall allow patients to upload and organize  medical documents. | Core | Planned |
| 4 | Doctors shall be able to issue e-prescriptions. | Core | Planned |
| 5 | The system shall support appointment booking,  rescheduling, and cancellation. | Core | Planned |
| 6 | The system shall send automatic reminders for appointments  and medications. | Utility | Planned |
| 7 | The system shall enable secure cloud storage of medical  data. | Enhancement | Planned |
| 8 | The system shall be able to share reports with doctors. | Core | Planned |
| 9 | The system shall provide real-time chat, and voice  consultation options. | Interactive | Planned |

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | The system shall synchronize user data across devices  (Smartphones, tablets, desktops). | Compatibility | Planned |

# **Non-Functional Requirements**

Table 3 Non-functional Requirements

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Non-Functional Requirements** | **Category** |
| 1 | The system shall provide end-to-end encryption for all patient data  and communication. | Security |
| 2 | The system shall support authentication mechanisms like passwords  and access tokens. | Security |
| 3 | The system shall comply with health data regulations | Security |
| 4 | The application shall maintain real-time synchronization of data  across all connected devices. | Performance |
| 5 | The user interface shall be intuitive, consistent, and accessible for all  user types (doctors and patients). | Usability |
| 6 | The system shall handle an increasing number of users and large  volumes of medical data efficiently. | Scalability |
| 7 | The system shall ensure availability and quick recovery through  automated data backups. | Reliability |
| 8 | The software shall allow for modular development and easy  maintenance using the Django framework. | Maintainability |

# **System Use Case Modeling**

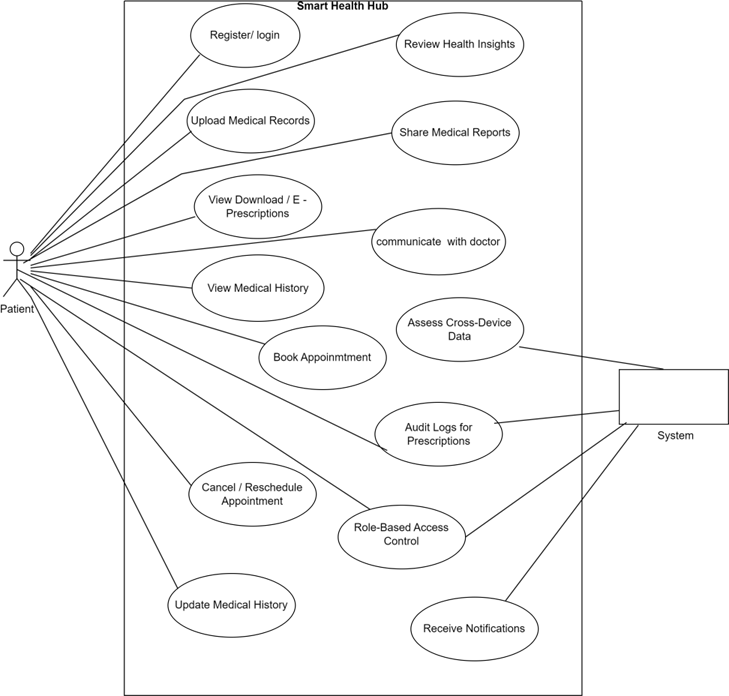


Figure 6 Patient Use Case Diagram

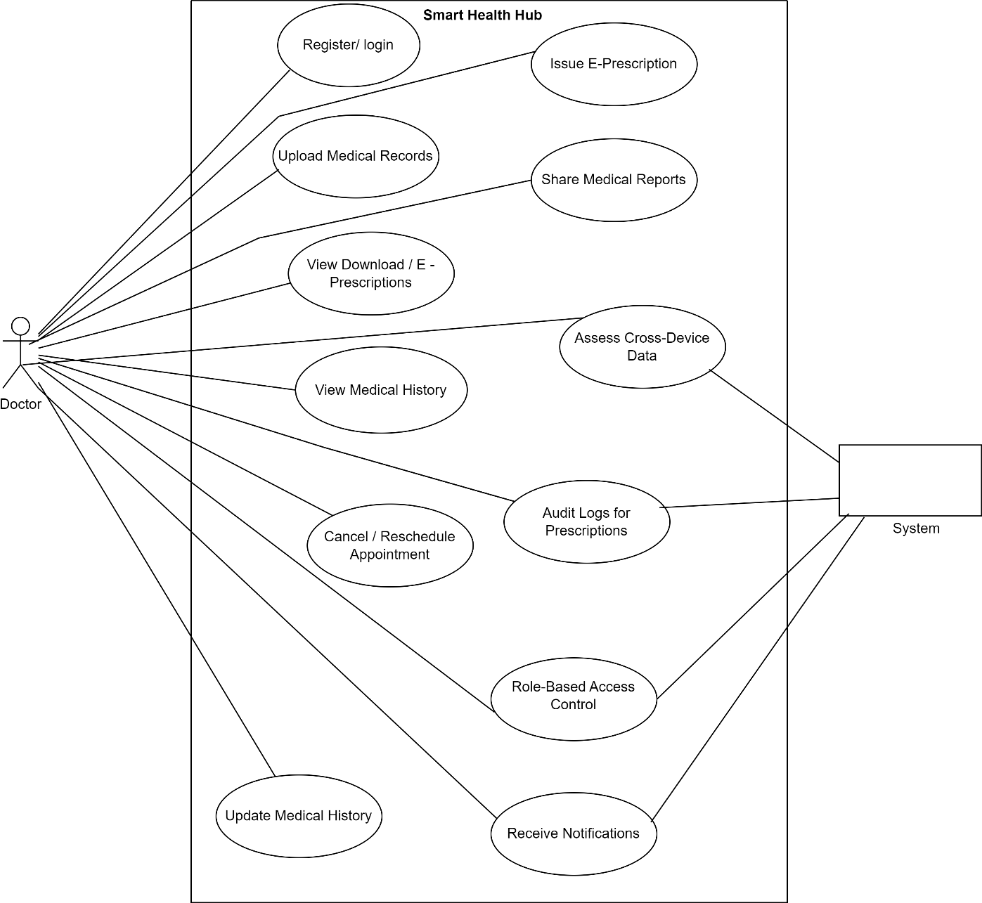


Figure 7 Doctor Use Case Diagram

## **Use Case 1: Register/Login**

Table 4 Register/Login

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-01 | | | | | | |
| **Use Case Name:** | Login/Registration | | | | | | |
| **Created By:** | Muhammad Talha Shafiq | | | **Last Updated By:** | | 13/04/2025 | |
| **Date Created:** | 10/04/2025 | | | **Last Revision**  **Date:** | |  | |
| **Actors:** | | | Patient, Doctor | | | | |
| **Description:** | | | Allows users to register a new account or log in to an existing one to access their health records and services. | | | | |
| **Trigger:** | | | User opens the application and clicks on Login/Register. | | | | |
| **Preconditions:** | | | The user has the application installed.  Internet access is available. | | | | |
| **Post conditions:** | | | User is logged in and can access their personal health data and features. | | | | |
| **Normal Flow:** | | | Actor | | System: | | |
| Opens the application  Clicks Login/Register  Enters credentials  Submits login details | | Displays login/register form  Validates input Grants access | | |
| **Alternative Flows:** | | | Forgot Password → Redirect to reset screen  New user → Redirect to registration form | | | | |
| **Exceptions:** | | | Incorrect credentials → Show error message  Network error → Show retry option | | | | |

## **Use Case 2: Upload Medical Record**

Table 5 Upload Medical Record

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-02 | | | | |
| **Use Case Name:** | Upload Medical Reports | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** | 14/04/2025 | | **Last Revision Date:** | |  |
| **Actors:** | | Patient | | | |
| **Description:** | | Allows patients to upload medical reports securely for record keeping and sharing. | | | |
| **Trigger:** | | User selects “Upload Report” option in the app. | | | |
| **Preconditions:** | | User is logged into the application  Internet connection is available | | | |
| **Post conditions:** | | Medical report is uploaded and saved in the patient’s account | | | |
| **Normal Flow:** | | Actor | | System | |
| Selects Upload Report  Selects files  Confirms upload | | Displays file picker  Uploads and saves the file  Displays success message | |
| **Alternative Flows:** | | Invalid file format → Show file format error | | | |
| **Exceptions:** | | Network issue → Retry or cancel option | | | |

## **Use Case 3: Book Appointment**

Table 6 Book Appointment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-03 | | | | |
| **Use Case Name:** | Book Appointment | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** | 15/04/2025 | | **Last Revision Date:** | |  |
| **Actors:** | | Patient | | | |
| **Description:** | | Allows patients to book an appointment with a doctor. | | | |
| **Trigger:** | | User selects “Book Appointment” in the application. | | | |
| **Preconditions:** | | User is logged into the system  Doctors are available with time slots | | | |
| **Post conditions:** | | Appointment booked and saved | | | |
| **Normal Flow:** | | Actor | | System | |
| Opens appointment booking  Selects date/time/doctor Confirms appointment | | Displays available time slots  Confirms availability  Saves booking, sends notification | |
| **Alternative Flows:** | | No available slots → Show message and alternative dates | | | |
| **Exceptions:** | | Network issue → Retry or cancel option | | | |

## **Use Case 4: Issue E-Prescription**

Table 7 Issue E-Prescription

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-04 | | | | |
| **Use Case Name:** | Issue e-Prescription | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** | 15/04/2025 | | **Last Revision Date:** | |  |
| **Actors:** | | Doctor | | | |
| **Description:** | | Allows a doctor to issue digital prescriptions for a patient. | | | |
| **Trigger:** | | Doctor selects a patient’s record and clicks “Issue Prescription.” | | | |
| **Preconditions:** | | Doctor is logged in  Patient record exists | | | |
| **Post conditions:** | | Prescription saved and sent to the patient | | | |
| **Normal Flow:** | | Actor | | System | |
| Opens patient’s profile  Enters prescription details Submits | | Displays patient information  Saves prescription  Sends prescription to patient | |
| **Alternative Flows:** | | Missing information → Show validation message | | | |
| **Exceptions:** | |  | | | |

## **Use Case 5: Communicate with Doctor**

Table 8 Communicate with Doctor

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-05 | | | | |
| **Use Case Name:** | Communicate with Doctor | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** | 15/04/2025 | | **Last Revision Date:** | |  |
| **Actors:** | | Patient, Doctor | | | |
| **Description:** | | Enables chat, voice communication between patient and doctor. | | | |
| **Trigger:** | | User clicks on “Chat” or “Consult” button | | | |
| **Preconditions:** | | User is logged in  Doctor is available online | | | |
| **Post conditions:** | | Communication is initiated | | | |
| **Normal Flow:** | | Actor | | System | |
| Selects doctor  Chooses chat/voice call Sends message or starts call | | Displays communication options  Initiates the selected option Relays message or connects call | |
| **Alternative Flows:** | | Doctor unavailable → Show unavailability message | | | |
| **Exceptions:** | | Connection error → Retry option | | | |

## **Use Case 6: Forgot Password**

Table 9 Forgot Password

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-06 | | | | |
| **Use Case Name:** | Forgot Password | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** | 15/04/2025 | | **Last Revision Date:** | |  |
| **Actors:** | | Patient, Doctor | | | |
| **Description:** | | Allows users to reset a forgotten password through verification. | | | |
| **Trigger:** | | User clicks “Forgot Password” on login page | | | |
| **Preconditions:** | | User email/phone is registered | | | |
| **Post conditions:** | | Password is reset | | | |
| **Normal Flow:** | | Actor | | System | |
| Clicks “Forgot Password”  Enters details Enters code  Submits | | Asks for email or phone number  Sends verification code Allows new password setting  Updates password | |
| **Alternative Flows:** | | Invalid contact info → Show error | | | |
| **Exceptions:** | | Network failure → Retry option | | | |

## **Use Case 7: Access Cross-Device Data**

Table 10 Access Cross Device Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-07 | | | | |
| **Use Case Name:** | Access Cross-Device Data | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** |  | | **Last Revision Date:** | |  |
| **Actors:** | | Patient, Doctor | | | |
| **Description:** | | Allows users to access data from different devices. | | | |
| **Trigger:** | | Login from a device | | | |
| **Preconditions:** | | Data synced to cloud | | | |
| **Post conditions:** | | Data available | | | |
| **Normal Flow:** | | Actor | | System | |
| logs in | | Fetches synced data | |
| **Alternative Flows:** | |  | | | |
| **Exceptions:** | | Data not found → Error | | | |

## **Use Case 8: View/Download E-Prescription**

Table 11 View/Download E-Prescription

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-08 | | | |
| **Use Case Name:** | View/Download E-Prescription | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** |  |
| **Date Created:** |  | | **Last Revision Date:** |  |
| **Actors:** | | Patient | | |
| **Description:** | | Enables patient to view or download electronic prescriptions. | | |
| **Trigger:** | | Patient selects prescription. | | |
| **Preconditions:** | | E-prescription exists. | | |
| **Post conditions:** | | Prescription downloaded | | |

|  |  |  |
| --- | --- | --- |
| **Normal Flow:** | Actor: Selects prescription | System: Displays details |
| Clicks download | Downloads file |
| **Alternative Flows:** |  | |
| **Exceptions:** | Download error | |

## **Use Case 9: View Medical History**

Table 12 View Medical History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-09 | | | | |
| **Use Case Name:** | View Medical History | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** |  | | **Last Revision Date:** | |  |
| **Actors:** | | Patient | | | |
| **Description:** | | Displays a summary of past health records. | | | |
| **Trigger:** | | Patient selects “Medical History.” | | | |
| **Preconditions:** | | Records available | | | |
| **Post conditions:** | | History displayed | | | |
| **Normal Flow:** | | Actor: | | System: | |
| Selects history | | Displays records | |
| **Alternative Flows:** | |  | | | |
| **Exceptions:** | | No records available | | | |

## **Use Case 10: Review Health Insights**

Table 13 Review Health Insights

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-10 | | | |
| **Use Case Name:** | Review Health Insights | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** |  |
| **Date Created:** |  | | **Last Revision Date:** |  |
| **Actors:** | | Patient | | |
| **Description:** | | Provides personalized health recommendations | | |
| **Trigger:** | | System generates insights | | |
| **Preconditions:** | | Data available | | |
| **Post conditions:** | | Insights displayed | | |

|  |  |  |
| --- | --- | --- |
| **Normal Flow:** | Actor: | System: |
| Analyzes data | Displays insights |
| **Alternative Flows:** |  | |
| **Exceptions:** | Insufficient data | |

## **Use Case 11: Update Medical History**

Table 14 Update Medical Records

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-11 | | | | |
| **Use Case Name:** | Update Medical History | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** |  | | **Last Revision Date:** | |  |
| **Actors:** | | Patient | | | |
| **Description:** | | Patient updates their health data | | | |
| **Trigger:** | | Selects update | | | |
| **Preconditions:** | | Logged in | | | |
| **Post conditions:** | | Data updated | | | |
| **Normal Flow:** | | Actor: | | System: | |
| Enters data | | Updates record | |
| **Alternative Flows:** | |  | | | |
| **Exceptions:** | | Validation error | | | |

## **Use Case 12: Audit Logs for Prescriptions**

Table 15 Audit Logs for Prescriptions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-12 | | | | |
| **Use Case Name:** | Audit Logs for Prescriptions | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** |  | | **Last Revision Date:** | |  |
| **Actors:** | | Doctor | | | |
| **Description:** | | Reviews log of issued prescriptions | | | |
| **Trigger:** | | Doctor selects logs | | | |
| **Preconditions:** | | Logs available | | | |
| **Post conditions:** | | Logs viewed | | | |
| **Normal Flow:** | | Actor | | System | |
| Selects logs | | Displays list | |
| **Alternative Flows:** | |  | | | |
| **Exceptions:** | | No logs | | | |

## **Use Case 13: Role Based Access Control**

Table 16 Role-Based Access Control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-13 | | | | |
| **Use Case Name:** | Role-Based Access Control | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** |  | | **Last Revision Date:** | |  |
| **Actors:** | | System | | | |
| **Description:** | | Manages data permissions based on roles | | | |
| **Trigger:** | | User logs in | | | |
| **Preconditions:** | | User authenticated | | | |
| **Post conditions:** | | Access granted/restricted | | | |
| **Normal Flow:** | | Actor: Grants/denies access | | System: Validates role | |
|  | |  | |
| **Alternative Flows:** | |  | | | |
| **Exceptions:** | | Invalid role | | | |

## **Use Case 14: Receive Notification**

Table 17 Receive Notifications

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-14 | | | | |
| **Use Case Name:** | Receive Notifications | | | | |
| **Created By:** | Muhammad Talha Shafiq | | **Last Updated By:** | |  |
| **Date Created:** |  | | **Last Revision Date:** | |  |
| **Actors:** | | Patient, Doctor | | | |
| **Description:** | | Sends alerts for appointments, prescriptions, medications | | | |
| **Trigger:** | | System event | | | |
| **Preconditions:** | | Notification service active | | | |
| **Post conditions:** | | Notification delivered | | | |
| **Normal Flow:** | | Actor | | System | |
| Displays alert | | Generates notification | |
| **Alternative Flows:** | |  | | | |
| **Exceptions:** | | Notification failure | | | |

# **System Sequence Diagram**

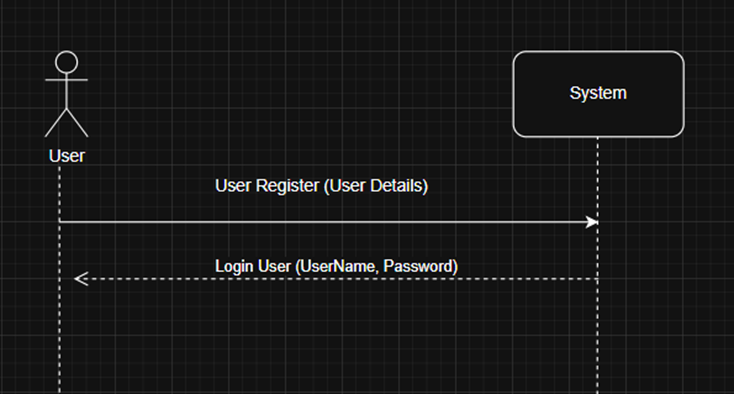
****

Figure 8 User Login/ Register

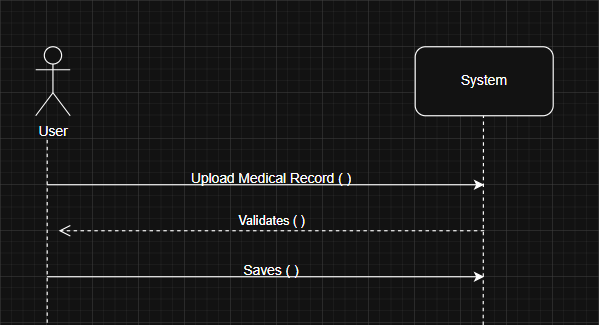


Figure 9 Upload Medical Reports

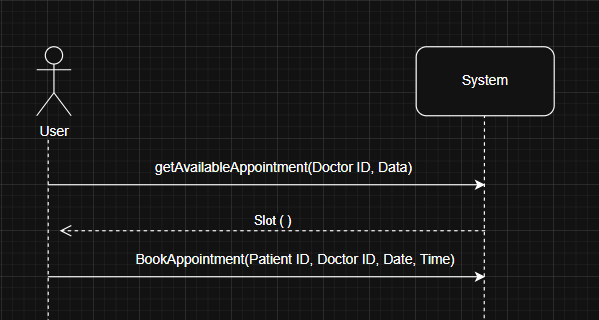
**

Figure 10 Book Appointment

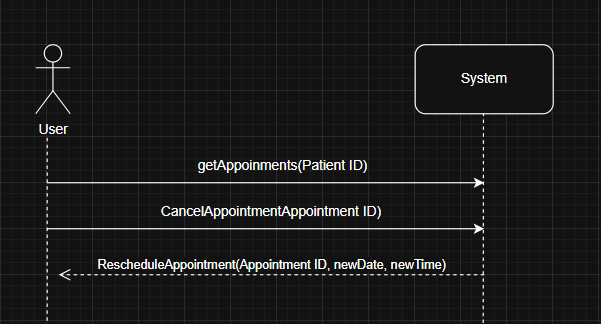
**

Figure 11 Reschedule/Cancel Appointments

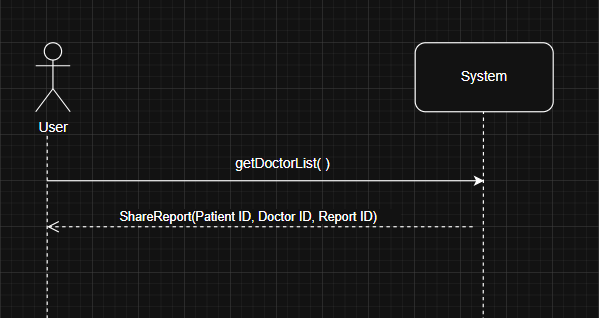


Figure 12 Share Reports

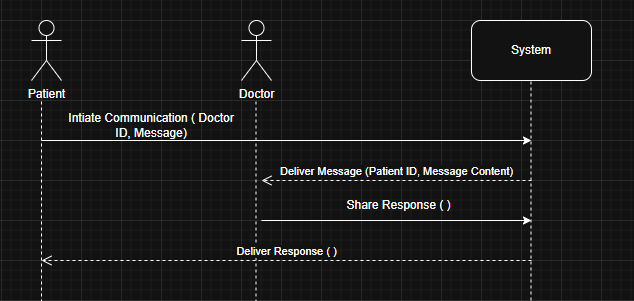


Figure 13 Communicate with Doctor

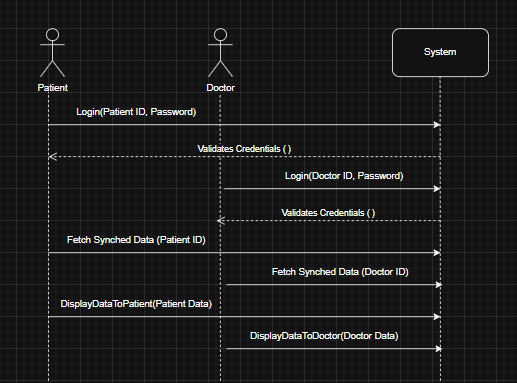
**

Figure 14 Access Cross Device Data

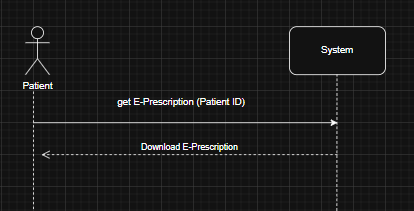


Figure 15 View/Download E-Prescription

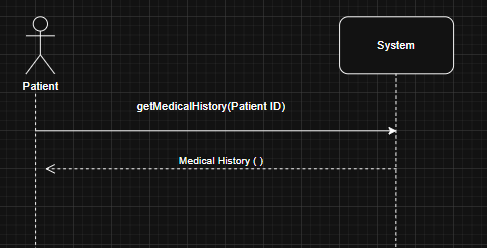
**

Figure 16 View/Download Medical History

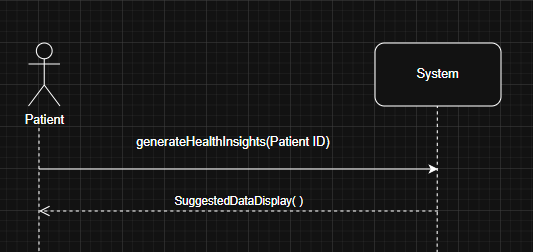
**

Figure 17 View Health Insights

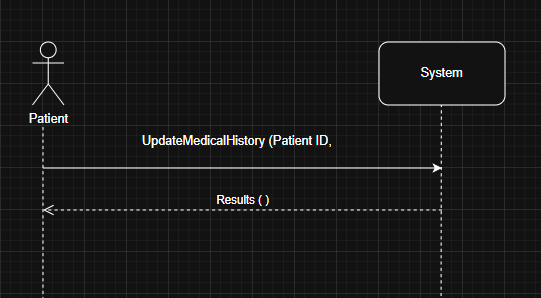


Figure 18 Update Medical History

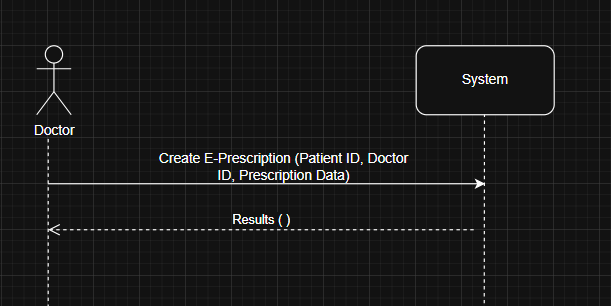
**

Figure 19 Issue E-Prescriptions

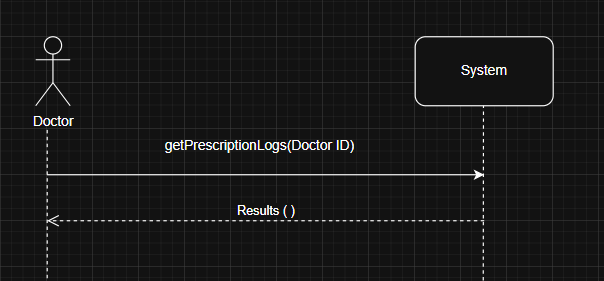
**

Figure 20 Audit Logs for Prescriptions

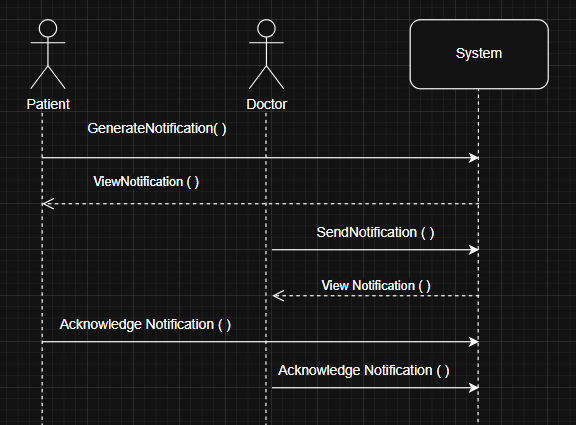


Figure 21 Receive Notifications

# **Domain Model**

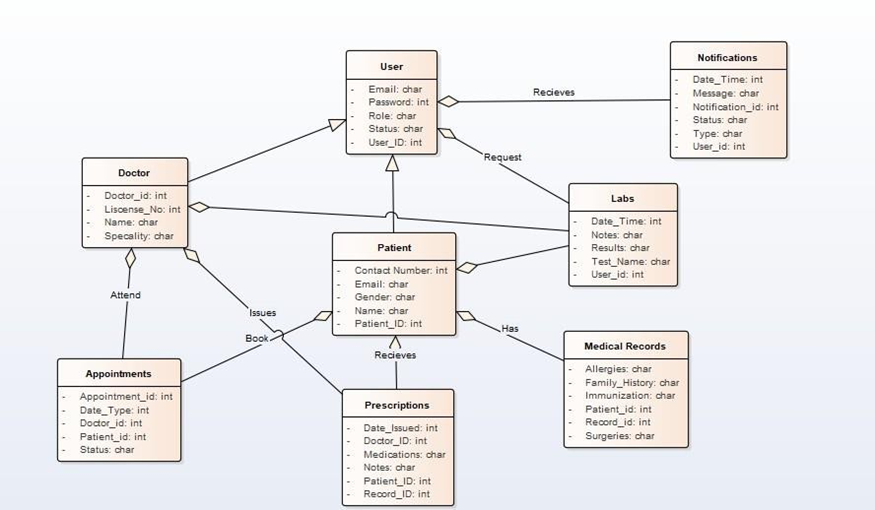


Figure 22 Domain Model

# **User Interface Design (Prototype)**

Smart Health Hub is designed to prioritize users’ health. The top bar offers sections like home services, Features, About Us, Contact Us and it also provides a personalized access for the users. This platform focuses on delivering modern healthcare management services directly to users' fingertips.

* + - Get Started button allows users to view the home page.
    - Log-In allows users to sign-in into their existing accounts.
    - Sign-Up allows users to sign-in into their existing accounts.

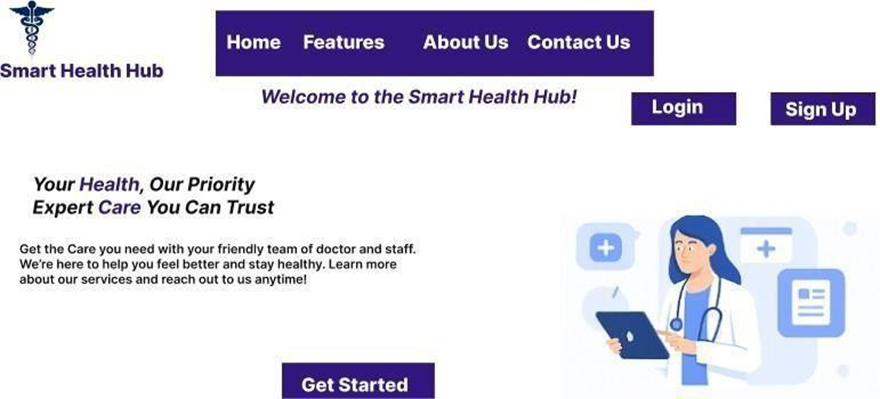


Figure 23 Home Screen

The users can sign in by adding their email and password

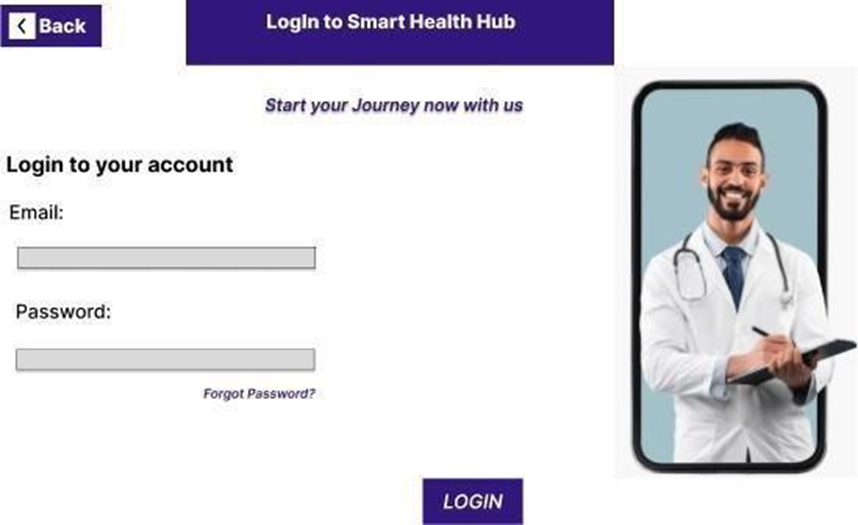


Figure 24 Login Screen

Users can sign -up by their email and setting up a password for their account.

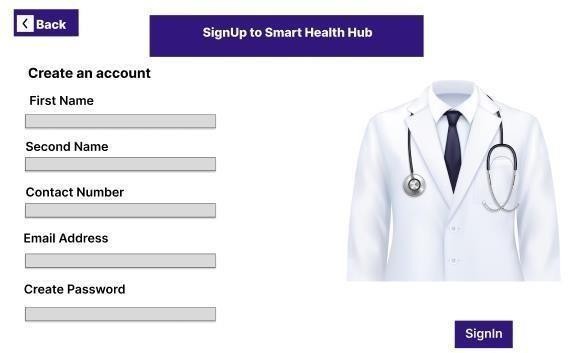


Figure 25 Registration Screen

Smart health hub offers six main essentials which are:

1. **ONLINE CONSULTATION**: This allows users to directly connect with healthcare professionals through chats or calls.
2. **ONLINE APPOINTMENTS**: This allows the users to easily book or reschedule their appointments.
3. **VIEW DOCTOR:** User can review doctors profile as they please
4. **VIEW HISTORY:** Users can access their history records anytime they are in need of it.
5. **SCHEDULE/CANCEL APPOINTMENT:** User can easily schedule or cancel their appointment
6. **MAINTAIN RECORD:** User can also maintain they record.



Figure 26 Feature Screen

This platform provides a secure and accessible digital healthcare solutions. That is easily accessible to everyone, everywhere.



Figure 27 About Screen

The users can contact us through: Name, Email, Message or Call Us:



Figure 28 Contact Us Screen

The Patient can also review their Dashboard respectively



Figure 29 Patient Dashboard Screen

The Patient can easily find a doctor according to disease or their choice respectively.

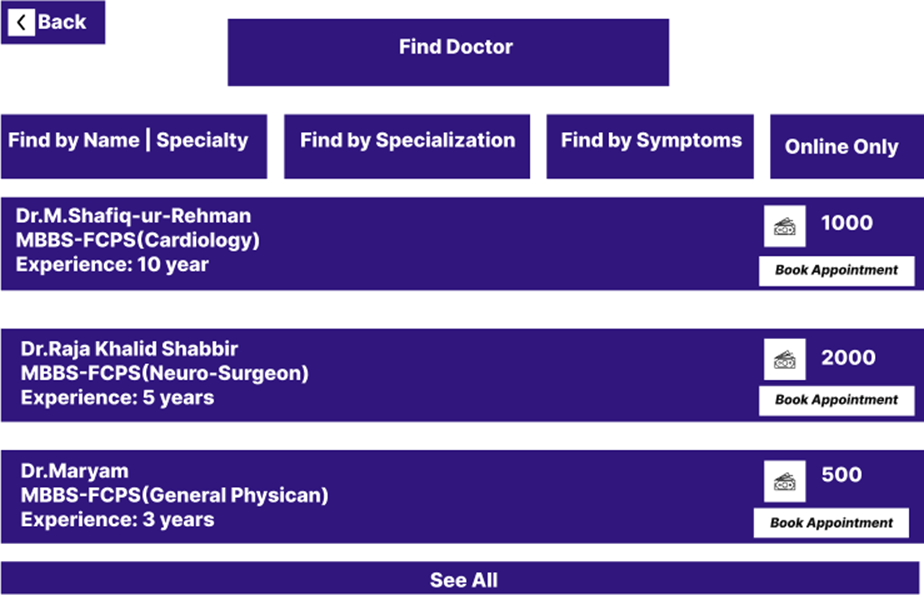


Figure 30 Find Doctor Screen

The Patient can schedule or cancel their appointments and can also review their past

appointments.



Figure 31 Schedule/Cancel Appointment Screen

The Patient can also review their labs.



Figure 32 Labs Screen

The Patient can also review and update their Medical History.

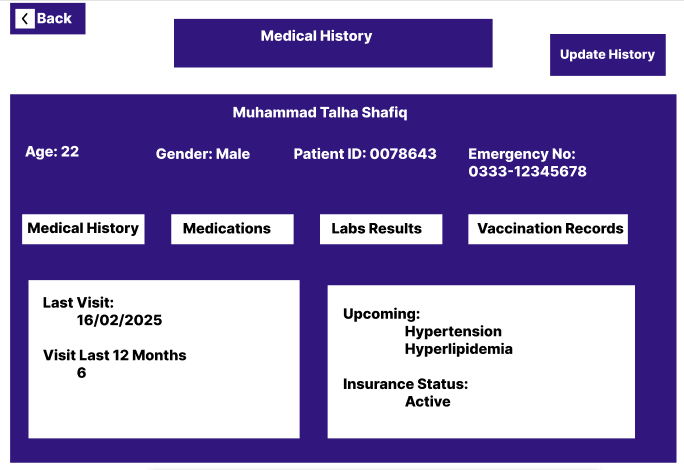


Figure 33 Medical History Screen

The Patients can easily get appointments

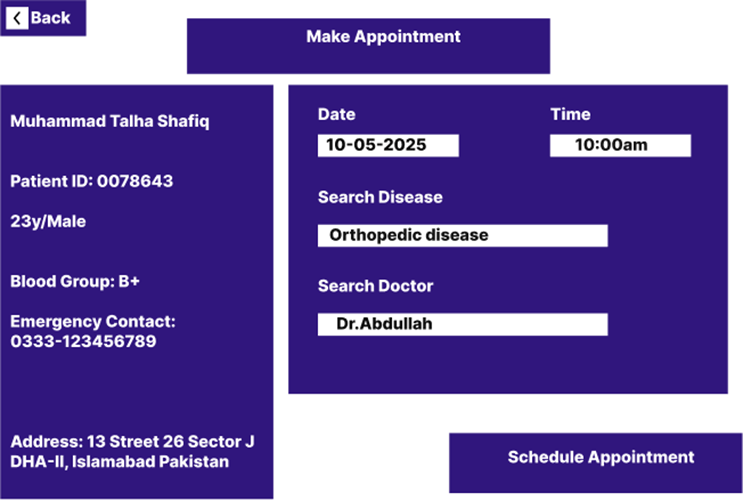


Figure 34 Make Appointment Screen

The Patients make and review their past payments with balance easily.

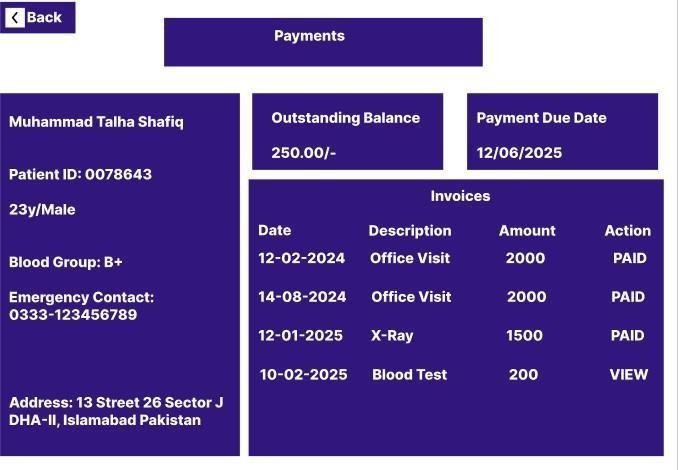


Figure 35 Payment Screen

The Doctors can review their Dashboard after login.



Figure 36 Doctor Dashboard Screen

The Doctors can easily access Patients Info’s with their past histories and appointments.

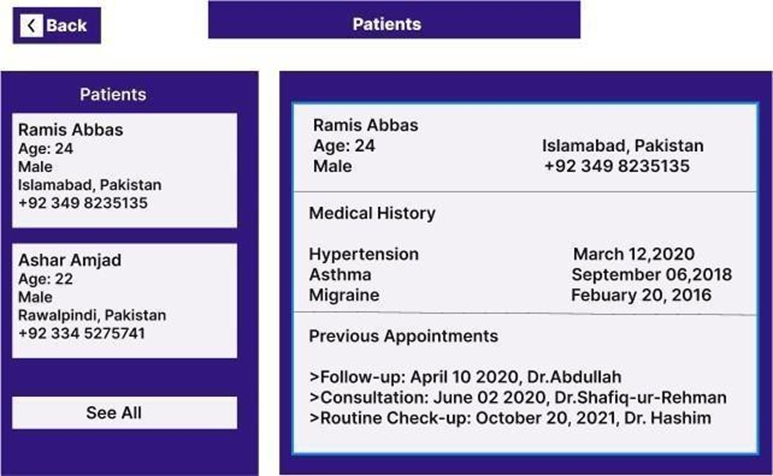


Figure 37 Patient Info Screen

The Doctors can see their Daily Appointments easily.



Figure 38 Appointment Screen

The Doctors have access to change their settings.

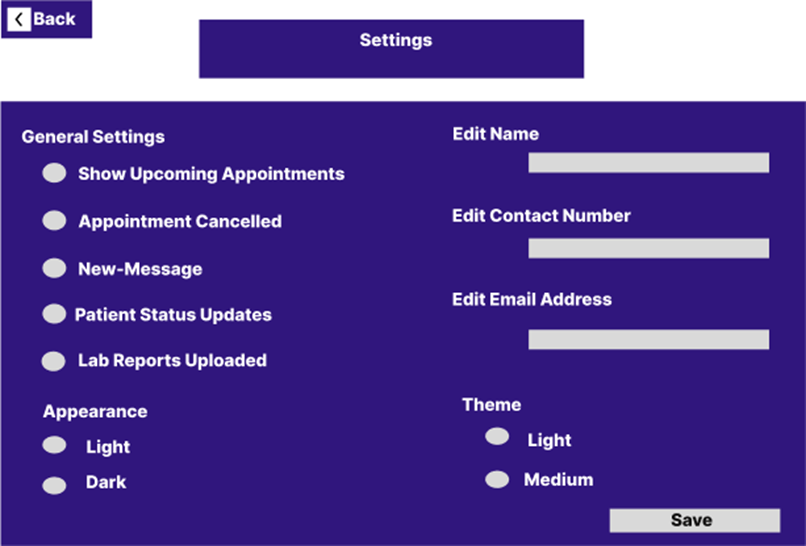


Figure 39 Settings Screen

Link: https:/[/www.figma.com/design/76A63WLNxnszTKLmq6R4xb/FYP?node-id=83-](http://www.figma.com/design/76A63WLNxnszTKLmq6R4xb/FYP?node-id=83-) 163&t=4K2Ivw3OxDbPohtT-0

# **Chapter 3**

# **System Design**

# **Software Architecture Diagram**

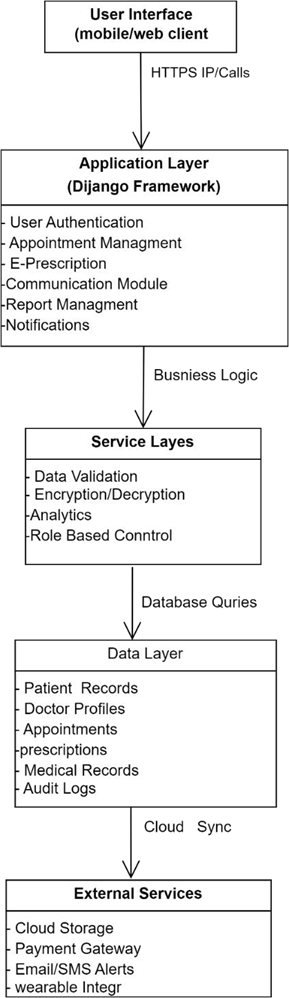


Figure 40 Software Architecture Diagram

# **Class Diagram**

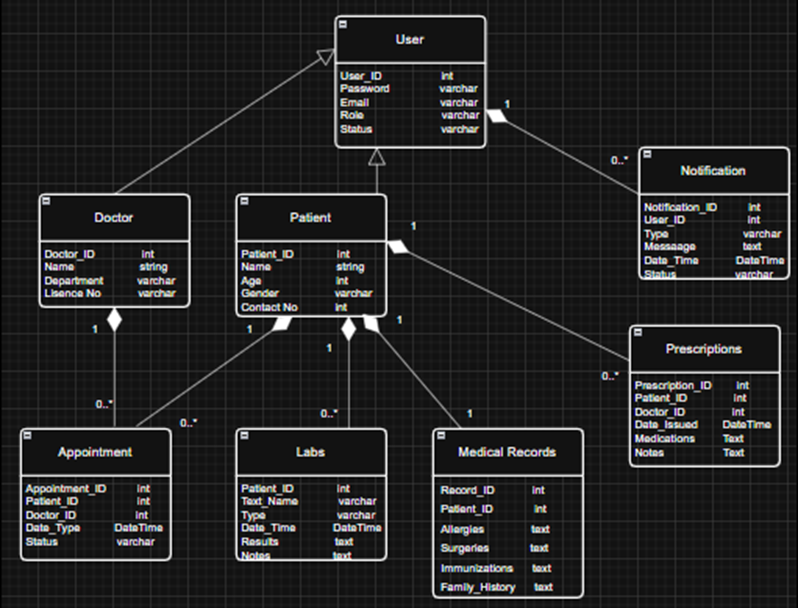


Figure 41 Class Diagram

# **Sequence Diagram**

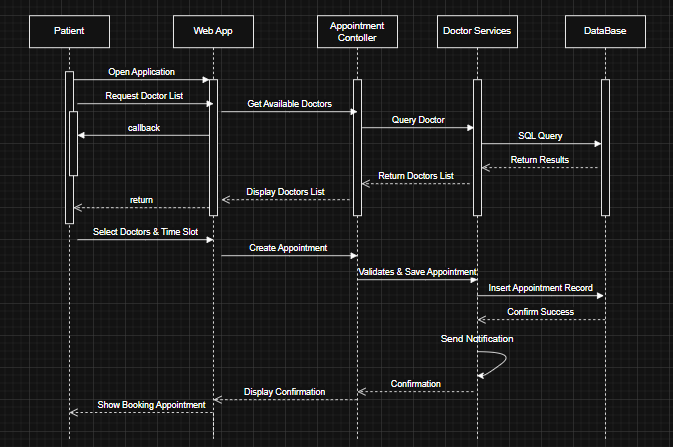


Figure 42 Sequence Diagram

# **Entity Relationship Diagram**

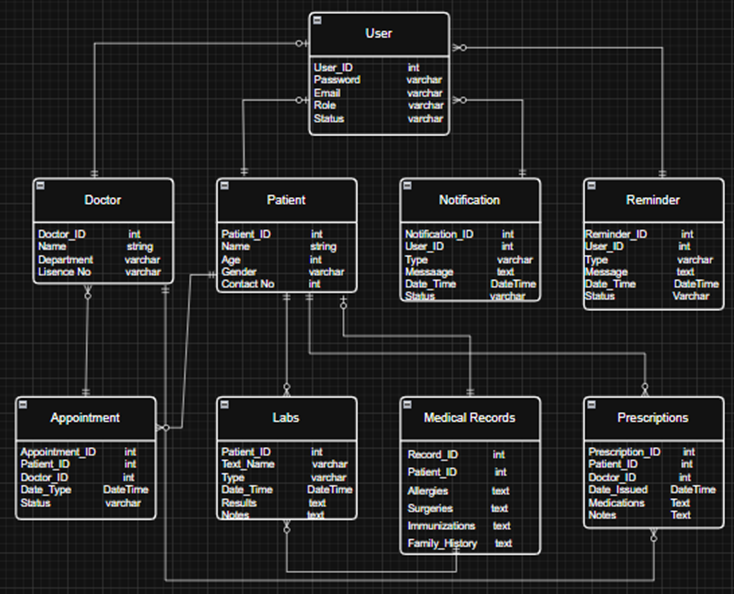


Figure 43 Entity Relationship Diagram

# **Database Schema**

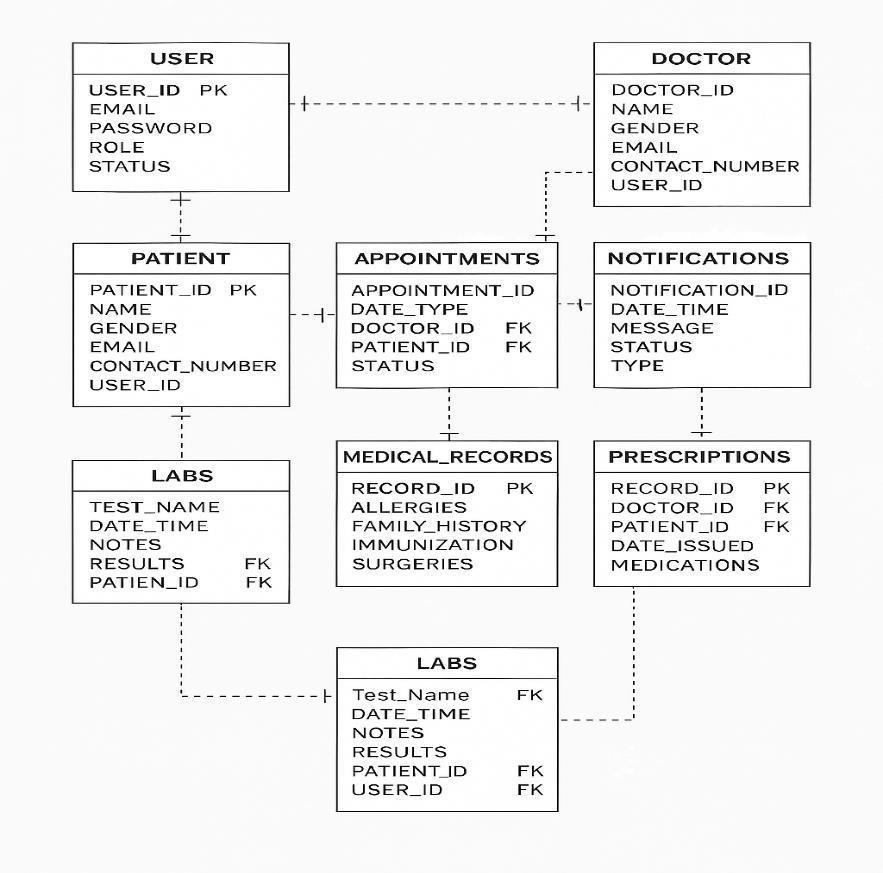


Figure 44 Database Schema

# **Chapter 4**

# **Software Description**

# **Coding Standards**

Coding standards are critical to maintaining clean, readable, and scalable code throughout the *Medilog* web application project. By following a unified coding style, we ensure better collaboration among developers and ease future enhancements.

**Naming Conventions:**

We adopted the following conventions across all frontend and backend components:

* PascalCase is used for classes and interface names (e.g., UserRecord, IPatientService).
* camelCase is used for variables, functions, and public properties (e.g., fetchPatientData, recordList, updateProfile()).
* Private properties use camelCase with a leading underscore (e.g., \_patientId).
* Component files and modules are named using PascalCase or kebab-case, depending on the framework used.
* Constants are written in UPPER\_SNAKE\_CASE (e.g., API\_BASE\_URL, MAX\_FILE\_SIZE).

# **Development Environment**

The *Medilog* web application is developed using a robust and scalable tech stack designed to offer a seamless user experience while securely managing personal health records.

**Frontend:**

* Developed using **React.js**, enabling a dynamic and responsive interface.
* **Tailwind CSS** is used for utility-first styling to ensure consistency and modern design.
* **React Router** handles navigation and routing.
* **Axios** is used for handling HTTP requests to the backend.

**Backend:**

* Built using **Node.js** with **Express.js** framework for building RESTful APIs.
* **MongoDB** is used as the primary NoSQL database for storing user profiles, health data, and medical history.
* **Mongoose** is used for schema definition and interaction with the MongoDB database.

**Authentication & Security:**

* **JWT (JSON Web Token)** is used for secure user authentication and authorization.
* **Helmet.js** and **CORS** are integrated for securing HTTP headers and managing cross-origin requests.

**Tools & Workflow:**

* Git and GitHub are used for version control and team collaboration.
* Postman is used for API testing and documentation.
* Figma was used during the UI/UX design phase to prototype the user interface.
* Dotenv is used to manage environment-specific configuration variables securely.

**Project Management & Deployment:**

* The app follows a modular folder structure for scalability and maintainability.
* Deployed on Render or Verce**l** for frontend and Railway or Render for backend.
* Environment variables are kept separate for development and production environments.

This setup ensures that Medilog is maintainable, secure, and optimized for a smooth user experience across different devices and browsers.

# **Software Description**

## **Registration**

def register\_user(data):

    db = Database()

    conn = db.connect()

    if not conn: return False

    try:

        cur = conn.cursor()

        cur.execute("INSERT INTO users (email, password\_hash, role) VALUES (%s, %s, %s) RETURNING user\_id",)(data["email"], hash\_password(data["password"]), data["role"])

        user\_id = cur.fetchone()[0]

        if data["role"] == "patient":

            cur.execute(

                "INSERT INTO patients (user\_id, name, phone\_number, date\_of\_birth) VALUES (%s, %s, %s, %s)",

                (user\_id, data["name"], data["phone"], data.get("dob"))

            )

        else:

            cur.execute(

                "INSERT INTO doctors (user\_id, name, phone\_number, department, specialization, license\_number) VALUES (%s, %s, %s, %s, %s, %s)",

                (user\_id, data["name"], data["phone"], data.get("department"), data.get("specialization"), data.get("license\_number"))

            )

        conn.commit() send\_welcome\_email(data["email"], data["name"])

        st.success("Registration Sucessfull")

        return True

    except Exception as e:

        print(f"Registration error: {e}")

        conn.rollback()

        return False

    finally:

        cur.close()

        conn.close()

## **Login**

def authenticate\_user(email, password):

    db = Database()

    conn = db.connect()

    if not conn:

        return None

    try:

        cur = conn.cursor()

        cur.execute("SELECT user\_id, email, password\_hash, role FROM users WHERE email = %s", (email,))

        user = cur.fetchone()

## **Patient Dashboard**

def show\_dashboard():

"""Display the patient dashboard with upcoming appointments and profile overview."""

    user = get\_current\_user()

    if not user or user["role"] != "patient":

        st.error("Please log in as a patient to access this page.")

        st.session\_state.page = "auth"

        st.rerun()

        return

    selected = patient\_sidebar()

    if selected == "Logout":

        st.session\_state.page = "logout"

        st.rerun()

        return

    elif selected != "Dashboard":

        st.session\_state.page = selected.lower().replace(" ", "\_")

        st.rerun()

        return

    st.title("Patient Dashboard")

    st.subheader(f"Welcome, {user['email']}!")

    db = Database()

    conn = db.connect()

    if not conn:

        st.error("Failed to connect to database.")

        return

    try:

        cur = conn.cursor()

        cur.execute(

            """

            SELECT p.name, p.phone\_number, p.date\_of\_birth

            FROM patients p

            JOIN users u ON p.user\_id = u.user\_id

            WHERE u.email = %s

            """,

            (user["email"],)

        )

        patient = cur.fetchone()

        st.write("### Profile Overview")

        cols = st.columns(3)

        if patient:

            name, phone, dob = patient

            with cols[0]:

                st.metric("Name", name)

            with cols[1]:

                st.metric("Phone", phone or "Not provided")

            with cols[2]:

                st.metric("Date of Birth", dob or "Not provided")

            profile\_completeness = 100 if all([name, phone, dob]) else 66 if name and (phone or dob) else 33

                st.metric("Profile Completeness", f"{profile\_completeness}%")

        else:

            st.warning("Profile details not found. Please update your profile.")

        cur.execute(

            """

            SELECT a.appointment\_id, a.appointment\_date, d.name, a.status

            FROM appointments a

            JOIN patients p ON a.patient\_id = p.patient\_id

            JOIN doctors d ON a.doctor\_id = d.doctor\_id

            JOIN users u ON p.user\_id = u.user\_id

            WHERE u.email = %s

            ORDER BY a.appointment\_date

            """,

            (user["email"],)

        )

        appointments = cur.fetchall()

        if appointments:

            st.write("### Appointment Overview")

            appointment\_data = [

                {"Appointment ID": appt[0], "Date": appt[1].strftime("%Y-%m-%d %H:%M"),

                "Doctor": appt[2], "Status": appt[3]}

                for appt in appointments

            ]

            df = pd.DataFrame(appointment\_data)

            total\_appointments = len(appointments)

            scheduled = len([appt for appt in appointments if appt[3] == "scheduled"])

            cancelled = len([appt for appt in appointments if appt[3] == "cancelled"])

            cols = st.columns(3)

            with cols[0]:

                st.metric("Total Appointments", total\_appointments)

            with cols[1]:

                st.metric("Scheduled", scheduled)

            with cols[2]:

                st.metric("Cancelled", cancelled)

            st.write("#### Appointment Timeline")

            fig\_timeline = px.timeline(

                df,

                x\_start="Date",

                x\_end="Date",

                y="Doctor",

                title="Appointment Schedule",

                color="Status",

                color\_discrete\_map={"scheduled": "#636EFA", "cancelled": "#EF553B"}

            )

            st.plotly\_chart(fig\_timeline)

            st.write("#### Appointment Status Distribution")

            status\_counts = df["Status"].value\_counts().reset\_index()

            status\_counts.columns = ["Status", "Count"]

            fig\_pie = px.pie(status\_counts, names="Status", values="Count",

            title="Appointment Status")

            st.plotly\_chart(fig\_pie)

            st.write("#### Appointment Frequency by Day")

            df["Day"] = pd.to\_datetime(df["Date"]).dt.day\_name()

            day\_counts = df["Day"].value\_counts().reset\_index()

            day\_counts.columns = ["Day", "Count"]

            fig\_bar = px.bar(day\_counts, x="Day", y="Count", title="Appointments by Day")

            st.plotly\_chart(fig\_bar)

        else:

            st.info("No appointments found.")

    except Exception as e:

        st.error(f"Error fetching data: {e}")

    finally:

        cur.close()

        conn.close()

## **Doctor Dashboard**

def show\_dashboard():

    """Display the doctor dashboard with upcoming appointments."""

    user = get\_current\_user()

    if not user or user["role"] != "doctor":

        st.error("Please log in as a doctor to access this page.")

        st.session\_state.page = "auth"

        st.rerun()

        return

    selected = doctor\_sidebar()

    if selected == "Logout":

        st.session\_state.page = "logout"

        st.rerun()

        return

    elif selected != "Dashboard":

        st.session\_state.page = selected.lower().replace(" ", "\_")

        st.rerun()

        return

    st.title("Doctor Dashboard")

    st.subheader(f"Welcome, {user['email']}!")

    db = Database()

    conn = db.connect()

    if not conn:

        st.error("Failed to connect to database.")

        return

    try:

        cur = conn.cursor()

        cur.execute(

            """

            SELECT d.name, d.department, d.specialization, d.license\_number

            FROM doctors d

            JOIN users u ON d.user\_id = u.user\_id

            WHERE u.email = %s

            """,

            (user["email"],)

        )

        doctor = cur.fetchone()

        st.write("### Profile Overview")

        finally:

            cur.close()

            conn.close()

        cols = st.columns(4)

        if doctor:

            name, department, specialization, license = doctor

        with cols[0]:

            st.metric("Name", name)

        with cols[1]:

            st.metric("Department", department or "Not provided")

        with cols[2]:

            st.metric("Specialization", specialization or "Not provided")

        with cols[3]:

            st.metric("License Number", license)

        profile\_completeness = 100 if all([name, department, specialization, license])

        else 75 if name and (department or specialization or license) else 25

            st.metric("Profile Completeness", f"{profile\_completeness}%")

        else:

            st.warning("Profile details not found. Please update your profile.")

        cur.execute(

        """

        SELECT a.appointment\_id, a.appointment\_date, p.name, a.status

        FROM appointments a

        JOIN patients p ON a.patient\_id = p.patient\_id

        JOIN doctors d ON a.doctor\_id = d.doctor\_id

        JOIN users u ON d.user\_id = u.user\_id

        WHERE u.email = %s

        ORDER BY a.appointment\_date

        """,

        (user["email"],)

        )

        appointments = cur.fetchall()

        if appointments:

        st.write("### Appointment Overview")

        appointment\_data = [

            {"Appointment ID": appt[0], "Date": appt[1].strftime("%Y-%m-%d %H:%M"),

            "Patient": appt[2], "Status": appt[3]}

            for appt in appointments

        ]

        df = pd.DataFrame(appointment\_data)

        total\_appointments = len(appointments)

        scheduled = len([appt for appt in appointments if appt[3] == "scheduled"])

        cancelled = len([appt for appt in appointments if appt[3] == "cancelled"])

        cols = st.columns(3)

        with cols[0]:

            st.metric("Total Appointments", total\_appointments)

        with cols[1]:

            st.metric("Scheduled", scheduled)

        with cols[2]:

            st.metric("Cancelled", cancelled)

        st.write("#### Appointment Timeline")

        fig\_timeline = px.timeline(

            df,

            x\_start="Date",

            x\_end="Date",

            y="Patient",

            title="Appointment Schedule",

            color="Status",

            color\_discrete\_map={"scheduled": "#636EFA", "cancelled": "#EF553B"}

        )

        st.plotly\_chart(fig\_timeline)

        st.write("#### Appointment Status Distribution")

        status\_counts = df["Status"].value\_counts().reset\_index()

        status\_counts.columns = ["Status", "Count"]

        fig\_pie = px.pie(status\_counts, names="Status", values="Count",

        title="Appointment Status")

        st.plotly\_chart(fig\_pie)

        st.write("#### Appointment Frequency by Day")

        df["Day"] = pd.to\_datetime(df["Date"]).dt.day\_name()

        day\_counts = df["Day"].value\_counts().reset\_index()

        day\_counts.columns = ["Day", "Count"]

        fig\_bar = px.bar(day\_counts, x="Day", y="Count", title="Appointments by Day")

        st.plotly\_chart(fig\_bar)

        else:

            st.info("No appointments found.")

    except Exception as e:

        st.error(f"Error fetching data: {e}")

    finally:

        cur.close()

        conn.close()

# **Chapter 5**

# **Software Testing**

# **Testing Methodology**

The testing methodology adopted for Medi-Log: Smart Health Hub is a combination of **Black Box Testing**, **White Box Testing**, and **User Acceptance Testing (UAT)**.

* **Black Box Testing**: This method was used to verify that the application meets the specified requirements without considering internal code structure. Testers interacted with the user interface to validate functionalities such as login, dashboard operations, and appointment scheduling.
* **White Box Testing**: Applied during the development phase by the developers to validate internal logic, control flow, and error handling.
* **User Acceptance Testing (UAT)**: Conducted with real users including doctors and patients to ensure that the system fulfills real-world healthcare workflows and usability requirements.
* **Regression Testing**: Performed after every update or bug fix to ensure new changes didn’t break existing features.

# **Testing Environment**

The Medi-Log: Smart Health Hub system was tested under the following environment:

* **Frontend**: HTML5, CSS3, JavaScript (React/Angular)
* **Backend**: Node.js / Django (Choose based on your implementation)
* **Database**: PostgreSQL / MySQL
* **Web Server**: Apache / Nginx
* **Operating System**: Windows 11 / Ubuntu 22.04
* **Browsers Tested**: Google Chrome, Mozilla Firefox, Microsoft Edge
* **Device Types**: Desktop, Laptop, Android Smartphone

# **Test Cases**

## **Test Case 1: User Login with Valid Credentials**

**Test Case Description:**

Verifies that a registered user (either doctor or patient) can successfully log in to the Medilog system using valid credentials.

**How Test Case Was Generated:**

Entered a registered email address and the correct password on the Medilog login page, then clicked the "Login" button.

**Expected Result of the Test Case:**

The system validates the credentials, initiates a session, and redirects the user to their respective dashboard (Patient Dashboard or Doctor Dashboard).

**Actual Result of the Test Case:**

**Passed** – The user was successfully authenticated and redirected to their designated dashboard.

Table 18 User Login with Valid Credentials

|  |  |
| --- | --- |
| Date: 24 June 2025 |  |
| *System: Medi-Log The Smart Health Hub* |  |
| *Objective:* User Login with Validated Credentials | *Test ID:*1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*  Enter Valid Email and Password | |
| *Expected Result:* Access granted; redirected to Dash-Board | |
| *Actual Result:* passed | |

## **Test Case 2: User Login with Invalid Credentials**

**Test Case Description:**

Verifies that the system prevents access when a user attempts to log in with incorrect credentials.

**How Test Case Was Generated:**

Entered a registered email with an incorrect password (or an unregistered email) on the Medi-log login screen and clicked the **"Login"** button.

**Expected Result of the Test Case:**

The system detects invalid credentials and displays an error message such as **"Invalid email or password"**, without granting access or redirecting to the dashboard.

**Actual Result of the Test Case:**

**Passed** – The login attempt was blocked, and an appropriate error message was shown to the user.

Table 19 User Login with Invalid Credentials

|  |  |
| --- | --- |
| Date: 24 June 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective:* User Login with Invalid Credentials | *Test ID:*1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*  Enter Invalid Email and Password | |
| *Expected Result:* Invalid credentials; stay on login screen | |
| *Actual Result:* passed | |

## **Test Case 3: Patient Registration**

**Test Case Description:**

Verifies that a new patient can successfully register an account through the Medilog registration form.

**How Test Case Was Generated:**

On the **Registration** page, a new patient filled out all the required fields (full name, email, password, phone number, etc.) and clicked the **"Register"** button.

**Expected Result of the Test Case:**

The system validates all input fields, checks for duplicate email, saves the information in the database, and displays a confirmation message like **"Registration successful. You may now log in."**

**Actual Result of the Test Case:**

**Passed** – The patient account was successfully created, and a confirmation message was displayed.

Table 20 Patient Registration

|  |  |
| --- | --- |
| Date: 24 June 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: Patient Registration* | *Test ID:*1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*  Enter Valid Email, Password, Contact Number | |
| *Expected Result:* Registration successful; redirect to Patient Dashboard | |
| *Actual Result:* passed | |

## **Test Case 4: Appointment Booking**

**Test Case Description:**

Verifies that a patient can successfully book an appointment with a doctor through the Medilog system.

**How Test Case Was Generated:**

Logged in as a patient, navigated to the **"Book Appointment"** section, selected a doctor, chose a date and time slot, added a short note/symptom, and clicked the **"Confirm Appointment"** button.

**Expected Result of the Test Case:**

The system checks doctor availability, validates all inputs, saves the appointment details to the database, and displays a confirmation message like **"Appointment successfully booked."** A notification is also sent to the selected doctor.

**Actual Result of the Test Case:**

**Passed** – The appointment was successfully created, confirmation was shown, and the doctor received a notification.

Table 21 Appointment Booking

|  |  |
| --- | --- |
| Date: 24 June 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: Appointment Booking* | *Test ID:*1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*  Enter Valid Patient Name Select Doctor  Select Date and Time Slot | |
| *Expected Result:* Appointment booked successfully; confirmation shown | |
| *Actual Result:* passed | |

## **Test Case 5: View Medical Records**

**Test Case Description:**

Verifies that a logged-in patient can successfully view their medical records, including lab reports and prescriptions.

**How Test Case Was Generated:**

Logged in as a patient, navigated to the **"Medical Records"** section from the dashboard. Clicked on the desired record to view detailed information such as diagnosis, lab reports, and doctor’s prescriptions.

**Expected Result of the Test Case:**

The system retrieves the relevant medical records from the database and displays them clearly, grouped by visit/doctor/date, including download/view options for attached files.

**Actual Result of the Test Case:**

**Passed** – The patient's medical records were correctly fetched and displayed, with proper formatting and access to files.

Table 22 View Medical Record

|  |  |
| --- | --- |
| Date: 24 June 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Medical Record* | *Test ID:*1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*  Enter Valid Patient ID, Action = View Medical Record | |
| *Expected Result:* Display list of uploaded medical records with details (date, doctor, diagnosis, attachments) | |
| *Actual Result:* passed | |

## **Test Case 6: View Prescription**

**Test Case Description:**

Verifies that a logged-in patient can successfully view prescriptions issued by their doctor, including dosage instructions, frequency, and duration.

**How Test Case Was Generated:**

Logged in as a patient, navigated to the “Prescriptions” section from the dashboard. Selected a prescription entry to view detailed information such as prescribed medicines, dosage schedule, instructions, and doctor’s notes.

**Expected Result of the Test Case:**

The system retrieves all active and past prescriptions linked to the patient’s medical profile. The prescriptions should be displayed clearly, grouped by visit/doctor/date, including downloadable/viewable prescription files (if uploaded).

**Actual Result of the Test Case:**

**Passed** – The patient’s prescriptions were displayed correctly with accurate details, proper formatting, and access to any attached files.

Table 23 View Prescriptions

|  |  |
| --- | --- |
| Date: 15 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Prescription* | *Test ID:*2 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*  Enter Valid Patient ID, Action = View Prescription | |
| *Expected Result: Display list of prescriptions with details ( )* | |
| *Actual Result:* passed | |

## **Test Case 7: View Profile (Patient)**

**Test Case Description:**

Verifies that a logged-in patient can successfully view their own profile information, including personal details, contact information, demographics, and any linked medical data (e.g., allergies, chronic conditions, or emergency contact, if applicable).

**How Test Case Was Generated:**

Logged in as a patient and navigated to the “Profile” or “My Account” section from the dashboard. Accessed the profile summary page to review personal details such as full name, age/date of birth, gender, contact number, email, address, and profile photo (if uploaded). Also checked for proper display of editable fields and read-only medical information.

**Expected Result of the Test Case:**

The system retrieves the patient’s profile data accurately and displays it in a structured format. All fields should be clearly labeled, properly formatted, and match the data stored in the patient’s account. Any non-editable medical fields should be view-only, and edit functionality (if enabled for personal details) should be accessible without issues.

**Actual Result of the Test Case:**

**Passed** – The patient’s profile information was retrieved and displayed correctly. All fields were accurate, formatting was consistent, and any edit options functioned as expected.

Table 24 View Profile (Patient)

|  |  |
| --- | --- |
| Date: 16 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Profile* | *Test ID:* 1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |  |  | | --- | | Enter Valid Patient ID, Action = View Profile | | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *Display complete patient profile with accurate personal details and medical information (view-only fields)* | | |
| *Actual Result:* passed | |

## **Test Case 8: Chat with Doctor**

**Test Case Description:**

This test case verifies that a logged-in patient can successfully access the chat module, view previous chat history, and send/receive messages with a doctor through the Patient Portal. It ensures that the chat interface loads properly, messages appear in the correct format, and interactions behave as expected.

**How the Test Case Was Generated:**

1. Logged into the system as a **patient**.
2. Navigated to the **“Chats”** or **communication** section from the dashboard.
3. Selected an existing chat labeled with the doctor’s name (as shown in the screenshot).
4. Observed the loaded chat window, including:
   * Doctor’s greeting messages
   * Patient’s sent messages
   * Chat layout (bubbles, timestamps, sender/receiver alignment)
5. Sent a new message (“Hello”) to verify message delivery.
6. Monitored UI for:
   * Display of the newly sent message
   * Proper alignment (patient messages on the right, doctor messages on the left)
   * Presence of text box and send button

**Expected Result**

1. The chat window should load successfully and display previous messages in correct order.
2. Doctor messages should appear on the left, and patient messages on the right (as visible in screenshot).
3. The patient should be able to type in the message field and send a message without issues.
4. The newly sent message should:
   * Display instantly in the chat window
   * Use consistent styling with previous patient messages
5. The chat list on the left should show the active conversation.

**Actual Result**

**Passed** – The patient–doctor chat function worked as expected:

* Previous chat messages loaded correctly.
* The doctor’s messages appeared on the left in white chat bubbles.
* Patient messages appeared on the right in blue chat bubbles.
* The message input box and send button were fully functional.
* The patient’s new message (“Hello”) appeared immediately and in the correct format.

Table 25 Chat with Doctor

|  |  |
| --- | --- |
| Date: 16 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: Chat with Doctor* | *Test ID:* 1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |  |  | | --- | | Logged in as patient, navigated to Chats, selected doctor chat, typed and sent message "Hello" | | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *Chat window loads showing previous messages; new message appears instantly in correct format and*  *alignment; patient can send and receive messages without errors* | | |
| *Actual Result:* passed | |

## **Test Case 9: View Dashboard (Patient)**

**Test Case Description:**

Verifies that a logged-in patient can successfully view the main dashboard, including profile information, appointment summaries, and visual graphs for better tracking of appointment trends.

**How Test Case Was Generated:**

Logged in as a patient using valid credentials. Upon successful login, the system redirected to the patient dashboard. Observed the following:

* + - * + Profile Overview section (Name, Phone, Date of Birth)
        + Appointment Overview section (Total, Scheduled, Completed, Cancelled)
        + Graphical representations:
      * Appointment Schedule Graph
      * Appointment Status Graph
      * Appointments by Day Graph

**Expected Result of the Test Case:**

The dashboard should load all sections correctly without errors. All textual and graphical components must accurately reflect the patient’s data:

* Profile Overview displays correct personal details
* Appointment Overview counts match the actual appointments in the system
* Graphs render correctly, are readable, and show accurate data trends
* Layout and formatting are consistent and responsive

**Actual Result of the Test Case:**

**Passed** – All dashboard sections, including graphs and overview panels, displayed correctly with accurate data and proper formatting.

Table 26 View Dashboard (Patient)

|  |  |
| --- | --- |
| Date: 16 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Dashboard-Patient* | *Test ID:* 1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |  |  | | --- | | Enter Valid Patient ID, Action = View Profile | | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *Display complete patient profile with accurate personal details and medical information* | | |
| *Actual Result:* passed | |

## **Test Case 10: View Dashboard (Doctor)**

**Test Case Description:**

Verifies that a logged-in doctor can successfully access and view the Doctor Dashboard, including profile details, appointment statistics, patient insights, and graphical reports for analyzing patient and appointment trends.

**How Test Case Was Generated:**

Logged in as a doctor using valid credentials. After successful authentication, the system redirected to the doctor dashboard. The following dashboard components were evaluated:

* **Profile Overview section** (Name, Specialization, Contact Information)
* **Appointment Overview section** (Total Appointments, Today’s Appointments, Upcoming, Completed)
* **Patient Insights section** (Total Patients, New Patients, Returning Patients)
* **Graphical representations:**
  + - * Daily Appointments Graph
      * Appointment Status Distribution Graph
      * Patients by Category Graph

**Expected Result of the Test Case:**

The dashboard should load completely without errors and must accurately display all doctor-specific information. Expected behavior includes:

* + - * **Profile Overview** shows correct doctor details
      * **Appointment Overview** numbers match actual appointment records
      * **Patient Insights** accurately reflect patient data linked to the doctor
      * **Graphs** load properly, are readable, and reflect true data patterns
      * **Overall dashboard layout** should be responsive, properly aligned, and visually consistent across all sections

**Actual Result of the Test Case:**

**Passed** – All dashboard sections, including doctor profile, appointment statistics, patient insights, and graphical components, displayed correctly with accurate data and proper formatting.

Table 27 View Doctor Dashboard

|  |  |
| --- | --- |
| Date: 16 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Dashboard-Doctor* | *Test ID:* 2 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |  |  | | --- | | Enter Valid Doctor ID, Action = View Profile | | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *Display complete doctor dashboard with accurate profile details, appointment statistics (total, today’s, upcoming, completed), patient insights, and correctly rendered graphical summaries.* | | |
| *Actual Result:* passed | |

## **Test Case 11: View Schedule**

**Test Case Description:**

Verifies that a logged-in doctor can successfully view the *Manage Schedule* section, including existing schedule slots, day-wise listings, start/end times, and schedule management options such as adding, updating, and deleting schedule entries.

**How Test Case Was Generated:**

Logged in as a doctor using valid credentials. After successful login, selected the **Schedule** option from the doctor side menu. The system loaded the **Manage Schedule** page. Observed the following elements:

* **Manage Schedule Heading**
* **Add Schedule Slots** button
* **Update Schedule** button
* **Delete Schedules** button
* **Schedule Table** displaying:
  + ID
  + Day (Monday, Tuesday, Wednesday, Thursday, Friday)
  + Start Time (displayed in milliseconds)
  + End Time (displayed in milliseconds)
* **Filter icon** for filtering rows
* Full list of schedule slots based on doctor availability

**Expected Result of the Test Case:**

The schedule page should load completely and function without errors:

* All existing **schedule slots** must be displayed accurately
* **Days** should match the saved schedule entries
* **Start Time** and **End Time** must correspond correctly to backend data
* **Add Schedule Slots** button should be visible and clickable
* **Update Schedule** button should allow modifying selected slot(s)
* **Delete Schedules** button should allow deleting selected slot(s)
* Table should load with proper formatting, column alignment, and responsiveness
* Filter option should work correctly if applied

**Actual Result of the Test Case:**

**Passed** – Schedule list loaded successfully with correct day entries, time values, and all schedule management buttons displayed and functioning properly.

Table 28 Manage Schedule

|  |  |
| --- | --- |
| Date: 16 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Schedule* | *Test ID:* 2 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |   Enter Valid Doctor ID, Action = View Schedule | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *Display complete doctor schedule with accurate day-wise slots, correct start and end time values visible options to add/update/delete schedules, and properly formatted schedule table with filtering support.* | | |
| *Actual Result:* passed | |

## **Test Case 12: Manage Appointments (Doctor)**

**Test Case Description**

Verifies that a logged-in doctor can successfully view the **Appointments** section, including the full list of booked appointments, their details (date, time slot, patient info, status, treatment, reference number), and available appointment-management actions such as selecting an appointment and canceling it.

**How the Test Case Was Generated**

Logged into the system using valid doctor credentials. After successful login, clicked on **Appointments** from the doctor sidebar menu. The system redirected to the **Appointments** page. Observed the following elements and behaviors:

**Visible Page Elements**

* **Appointments Heading**
* Full appointments table with the following columns:
  + Appointment ID
  + Date
  + Time Slot
  + Patient Name
  + Status (scheduled, completed, cancelled, etc.)
  + Reference Number
  + Treatment Type
  + Patient Email
* Pagination controls at the bottom
* A dropdown to **Select Appointment to Cancel**
* A **Cancel Appointment** button
* Row highlighting/selection for choosing an appointment to cancel
* Filter/search icon (if implemented)

**Functional Behaviors Observed**

* Table loads all appointment entries associated with the logged-in doctor
* Status and details align with backend values
* Selecting an appointment populates the dropdown for cancellation
* Cancel button is visible and clickable

**Expected Result of the Test Case**

The Appointment page should load fully and function without any errors:

**Table Data Accuracy**

All existing appointments must be displayed with correct:

* + - * + Appointment ID
        + Date
        + Time Slot
        + Patient Name
        + Status
        + Reference Number
        + Treatment type
        + Patient Email

Appointment order should follow system logic (e.g., by date or ID)

**UI and Functional Behavior**

* Page heading **Appointments** must appear clearly
* Table should render correctly with proper formatting and alignment
* Pagination (if present) should navigate properly
* Filter/search option should work as expected
* Selecting an appointment should correctly populate the “Select Appointment to Cancel” dropdown
* Cancel Appointment button should:
  + Be visible
  + Be clickable
  + Trigger a cancellation action for the selected appointment
  + Update the appointment’s status to **cancelled** upon success

**Responsiveness**

* Table should remain readable and scrollable
* Columns should remain aligned on different screen sizes

**Actual Result of the Test Case**

**Passed** – The Appointment page loaded correctly with all appointment entries displayed.

* All appointment fields (ID, date, time slot, status, etc.) appeared accurate
* Selection dropdown displayed the selected appointment’s ID
* Cancel Appointment button appeared and functioned correctly
* Table rendered smoothly with proper alignment and pagination
* No errors occurred during loading or cancellation

Table 29 View Appointments

|  |  |
| --- | --- |
| Date: 18 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Appointments* | *Test ID:* 2 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |   Enter Valid Doctor ID, Action = View Appointments | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *Display complete appointment list with accurate appointment IDs, dates, time slots, patient details, statuses, treatment types, and reference numbers. All management options (select appointment dropdown and cancel appointment button) must be visible, clickable, and functioning correctly. Table should load fully formatted with pagination and filtering support.* | | |
| *Actual Result:* passed | |

## **Test Case 13: Manage Treatment**

**Test Case Description**

Verifies that a logged-in doctor can successfully view the **Manage Treatments** section, including existing treatment entries, their details (ID, name, description, cost), and the available treatment-management options such as adding, updating, and deleting treatments.

**How Test Case Was Generated**

Logged into the system using valid doctor credentials. After successful login, selected the **Treatments** option from the doctor side menu. The system loaded the **Manage Treatments** page. Observed the following elements:

**Visible Elements on Page**

* **Manage Treatments** heading
* **Add New Treatment** button
* **Update Treatment** button
* **Delete Treatments** button
* **Treatment Table** displaying:
  + Treatment ID
  + Name
  + Description
  + Cost (PKR)
* **Filter icon** for filtering rows
* Full list of treatments added by the doctor

**Functional Behavior Observed**

* Treatments load correctly from the backend
* All treatment rows display consistent values
* Action buttons are visible and clickable
* Table formatting is aligned and responsive
* Pagination visible when record count increases

**Expected Result of the Test Case**

The **Manage Treatments** page should load completely and function without errors:

**Data Accuracy**

* All existing treatments must be displayed correctly
* Treatment IDs, names, descriptions, and costs must match backend data
* No missing or misaligned table values

**UI and Functional Requirements**

* **Add New Treatment** button should be visible and responsive
* **Update Treatment** button should allow modification of selected treatment(s)
* **Delete Treatments** button should allow deletion of selected treatment(s)
* Table should load with:
  + Correct formatting
  + Proper column alignment
  + Responsive layout
  + Pagination (if applicable)
* Filtering should work properly when applied

**Usability Requirements**

* Page should load without errors
* Buttons should be easily accessible and clearly labeled
* Table should remain scrollable and readable

**Actual Result of the Test Case**

**Passed** – Treatment list loaded successfully with accurate details, correct alignment, and fully functional treatment management buttons (Add, Update, Delete). Filtering and table structure also worked without errors.

Table 30 Manage Treatment

|  |  |
| --- | --- |
| Date: 18 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: Manage Treatment* | *Test ID:* 2 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |   Enter Valid Doctor ID, Action = View Treatments | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *Display complete treatment list with accurate treatment IDs, names, descriptions, and costs. All management options (Add New Treatment, Update Treatment, Delete Treatments) must be visible, clickable, and functioning correctly. Table should load fully formatted with proper alignment, pagination, and filtering support.* | | |
| *Actual Result:* passed | |

## **Test Case 14: View Prescription (Doctor)**

**Test Case Description**

Verifies that a logged-in doctor can successfully view all prescriptions created for their patients and generate a new prescription with correct medication details, dosage, and duration.

**How Test Case Was Generated**

Logged in as a doctor, navigated to the Prescriptions section. Reviewed the displayed prescription list, charts, and analytics. Clicked “Generate New Prescription”, selected a patient (MR Number), added medication details, and submitted the prescription.

**Expected Result of the Test Case**

The system displays all existing prescriptions with accurate information, grouped by patient/date. The doctor should be able to generate a new prescription, add one or more medications, and submit it successfully. The new prescription should appear in the table and reflect correctly in the analytics/charts.

**Actual Result of the Test Case**

**Passed** – All prescriptions were displayed correctly with proper formatting. The doctor successfully generated a new prescription, and the newly created entry appeared immediately in the prescription list and dashboard charts.

Table 31 View Prescription

|  |  |
| --- | --- |
| Date: 18 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Prescription* | *Test ID:* 2 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |   Enter Valid Doctor ID, Action = View Prescriptions | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *The system should display the complete list of prescriptions with accurate prescription IDs, patient names, medication details, dosage, duration, and status. All prescription management options (View Prescription Details, Generate New Prescription) must be visible, clickable, and functioning correctly. Charts such as Medication Distribution, Prescriptions by Patient, Prescriptions Over Time, and Prescription Status should load with correct data. The prescription table should load fully formatted with proper alignment, pagination, and sorting/filtering support.* | | |
| *Actual Result:* passed | |

## **Test Case 15: View Profile (Doctor)**

**Test Case Description**

Verifies that a logged-in doctor can successfully view and manage their own profile information, including personal details, professional credentials, contact information, and optional password update functionality.

**How Test Case Was Generated**

Logged in as a doctor and navigated to the “Profile” section from the left-side Doctor Menu. The system opened the Manage Profile page, displaying the doctor’s profile fields such as:

* + - Name
    - Phone Number
    - Department
    - Specialization
    - License Number
    - Email
    - New Password (optional update field)

Observed which fields were editable (e.g., name, phone number) and which were read-only (department, specialization, license number, email—based on screenshot shading). Checked visibility, formatting, and functionality of the Update Profile button. Confirmed password field behavior (optional and masked).

**Expected Result of the Test Case**

The system should retrieve and display the doctor’s profile accurately. All fields should:

* + - Be pre-filled with the correct stored values
    - Show proper formatting and labels
    - Display read-only fields in non-editable state (e.g., department, specialization, license number)
    - Allow editable fields to be updated without errors
    - Allow password to be changed only if a new value is entered
    - Successfully save updates when the Update Profile button is clicked
    - Display no unauthorized or irrelevant fields

**Actual Result of the Test Case**

**Passed** –The doctor’s profile loaded correctly, and all fields displayed as expected. Read-only sections were non-editable, editable fields were functioning correctly, and the password field behaved as intended (optional and hidden). The Update Profile action worked successfully without any issues.

Table 32 View Profile (Doctor)

|  |  |
| --- | --- |
| Date: 18 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: View Profile* | *Test ID:* 1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |  |  | | --- | | Enter Valid Doctor ID, Action = View Profile | | |
| *Expected Result:*   |  | | --- | |  |  |  | | --- | | *Display complete patient profile with accurate personal details and medical information (view-only fields)* | | |
| *Actual Result:* passed | |

## **Test Case 16: Chat with Patient**

**Test Case Description**

This test case verifies that a logged-in doctor can successfully open, view, and interact with an existing chat session with a patient. It ensures that the chat interface correctly displays patient details (name/avatar), chat history, and allows the doctor to send new messages. It also validates that previously stored messages appear in correct order and formatting.

**How the Test Case Was Generated**

* Logged into the system using a doctor account.
* Navigated to the Doctor Portal Dashboard.
* Selected a patient conversation from the “Recent Chats” section
* Opened the chat window to view:
  + Patient name and avatar displayed at the top.
  + Conversation history including messages from both doctor and patient.
  + Input box and send button for composing replies.
* Observed the message layout and verified message direction (doctor messages on right, patient on left).

**Expected Result**

The system should correctly display the patient’s name/avatar, load the complete chat history in proper order, and show patient messages on the left and doctor messages on the right. The doctor should be able to type and send new messages, which should appear instantly without errors or unauthorized data.

**Actual Result**

**Passed** – The chat interface displayed patient details and message history correctly, with proper alignment and formatting. All doctor messages sent successfully and appeared instantly, with no UI issues or data inconsistencies.

Table 33 Chat with Patient

|  |  |
| --- | --- |
| Date: 18 November 2025 |  |
| *System: Medi-Log: The Smart Health Hub* |  |
| *Objective: Chat with Patient* | *Test ID:* 1 |
| *Version:*1 | *Test Type:* Unit testing |
| *Input:*   |  | | --- | |  |   Open Doctor Portal → Select a patient from "Recent Chats" → Action = Open Chat Window | |
| *Expected Result:*   |  | | --- | |  |   Chat interface loads with patient name/avatar, complete message history in correct order, and proper left/right message alignment; doctor can type and send new messages successfully. | |
| *Actual Result:* passed | |