**Project Report**



**Doctor Management Module**

DEPARTMENT OF COMPUTER SCIENCE

**Dated: 15 May 2024**

# TABLE OF CONTENTS

Contents

[TABLE OF CONTENTS 2](#_Toc166709618)

[INTRODUCTION: 3](#_Toc166709619)

[Project Background: 5](#_Toc166709620)

[CHAPTER 01: UI / UX 16](#_Toc166709625)

[1.1 Requirements 16](#_Toc166709626)

[1.2 Color Palette 18](#_Toc166709627)

[1.3 Wireframes and Mockups 18](#_Toc166709628)

[CHAPTER 02: DATABASE 21](#_Toc166709629)

[2.1 Introduction 21](#_Toc166709630)

[2.2 Requirements 21](#_Toc166709631)

[2.3 ERD Description 23](#_Toc166709632)

[2.4 Diagrams 24](#_Toc166709633)

[2.5 Entities/Attributes 27](#_Toc166709634)

[CHAPTER 03: Web Application 28](#_Toc166709635)

[3.1 Introduction 28](#_Toc166709636)

[3.2 Requirements 28](#_Toc166709638)

[3.3 Functional Requirements 29](#_Toc166709639)

[3.4 Module Handling 33](#_Toc166709640)

[Chapter 4: BACKEND 34](#_Toc166709641)

[4.2 Function and Description](#_Toc166709642) 38

Chapter 5: USER GUIDE………………………………………………………………………………………………………………………….39

[CONCLUSION 47](#_Toc166709643)

## 

## INTRODUCTION:

In the ever-evolving landscape of healthcare, the efficient management of doctor-patient interactions and administrative tasks stands as a cornerstone for delivering quality care. Introducing the Doctor Management Module, a pivotal component within our comprehensive hospital management system. This module represents a concerted effort to redefine how doctors engage with patients, manage appointments, and access critical information within healthcare facilities.

Navigating the complexities of modern healthcare demands innovative solutions that streamline processes, enhance communication, and empower medical professionals. The Doctor Management Module emerges as a response to these challenges, offering a sophisticated platform designed to optimize workflow, elevate patient care, and foster operational excellence.

At its core, this module embodies a commitment to efficiency, effectiveness, and patient-centric care. By providing doctors with intuitive tools to manage their schedules, access patient records, and coordinate care, we aim to catalyze positive outcomes and elevate the standard of healthcare delivery.

Driven by a dedication to innovation and excellence, our project seeks to revolutionize the doctor-patient experience. Through seamless integration of advanced technology, user-friendly interfaces, and robust data management capabilities, the Doctor Management Module promises to redefine the way healthcare is delivered and experienced.

With a focus on collaboration, communication, and continuous improvement, our project represents more than just a software solution—it embodies a vision for a future where healthcare is accessible, efficient, and patient-centered. Join us on this journey as we embark on a mission to transform the healthcare landscape, one interaction at a time.

**Project Objective:**

The primary objective of the Doctor Management Module is to revolutionize the management of doctor-patient interactions and administrative tasks within healthcare facilities. This overarching goal is achieved through a series of specific objectives aimed at enhancing efficiency, improving patient care, and optimizing operational processes.

1. **Efficient Appointment Management:**
   * The module aims to streamline the scheduling and management of doctor appointments, allowing for easy coordination between doctors and patients.
   * Objective: Implement a user-friendly appointment scheduling system that minimizes scheduling conflicts and optimizes doctor availability.
2. **Access to Patient Information:**
   * Providing doctors with access to comprehensive patient information is crucial for delivering personalized and effective care.
   * Objective: Develop a centralized platform that allows doctors to access patient records, medical history, and relevant clinical data in real-time.
3. **Enhanced Communication and Coordination:**
   * Effective communication and coordination between doctors, patients, and administrative staff are essential for delivering seamless healthcare services.
   * Objective: Implement features that facilitate communication between doctors and patients, such as appointment reminders, messaging systems, and notifications.
4. **Optimized Workflow:**
   * By automating repetitive tasks and streamlining workflow processes, the module aims to improve operational efficiency within healthcare facilities.
   * Objective: Develop tools and functionalities that automate administrative tasks, such as appointment reminders, billing, and documentation.
5. **Improved Patient Care:**
   * The ultimate goal of the module is to enhance the quality of patient care by providing doctors with the tools and information they need to make informed decisions.
   * Objective: Empower doctors with access to comprehensive patient information, diagnostic tools, and decision support systems to facilitate accurate diagnoses and personalized treatment plans.
6. **Data Security and Compliance:**
   * Ensuring the security and confidentiality of patient data is paramount in healthcare settings, where sensitive information is handled on a daily basis.
   * Objective: Implement robust security measures and compliance protocols to protect patient data and ensure adherence to regulatory requirements, such as HIPAA.
7. **User Satisfaction and Adoption:**
   * The success of the module hinges on user satisfaction and adoption, both among doctors and patients.
   * Objective: Design an intuitive and user-friendly interface that promotes ease of use, encourages adoption, and enhances user satisfaction.
8. **Scalability and Future-Readiness:**
   * As healthcare needs evolve and grow, the module must be scalable and adaptable to accommodate future requirements.
   * Objective: Design an architecture that is scalable, flexible, and easily extensible to support future enhancements, updates, and expansions.

By addressing these specific objectives, the Doctor Management Module aims to transform the way healthcare is delivered, experienced, and managed within healthcare facilities, ultimately improving patient outcomes and enhancing the overall quality of care.

### Project Background:

The project background consists of the following sub-sections:

#### Purpose:

The purpose of the Doctor Management Module is to:

* Facilitate seamless communication and coordination among doctors, patients, and administrative staff
* Enhance the quality of patient care by providing doctors with access to relevant patient information and medical history
* Improve operational efficiency within the healthcare facility by automating appointment scheduling and administrative tasks

#### Scope:

1. **Secure Doctor Login Panel:**
   * This feature provides a secure authentication mechanism for doctors to access the system. It ensures that only authorized personnel can log in, safeguarding sensitive data and maintaining confidentiality. Authentication typically involves the use of unique credentials (e.g., username and password) and may incorporate additional security measures such as multi-factor authentication for added protection.
2. **Comprehensive Doctor Dashboard:**
   * The doctor dashboard serves as a centralized interface that presents relevant information and functionalities to doctors upon login. It provides an overview of upcoming appointments, patient interactions, notifications, and other essential data. The dashboard allows doctors to efficiently manage their workload, track their schedules, and access key features of the system without navigating through multiple screens.
3. **Detailed Doctor Profiles:**
   * Doctor profiles contain comprehensive information about each doctor within the healthcare facility. This includes personal details (e.g., name, contact information), professional qualifications (e.g., experience, specialization), work schedule, and profile picture. Detailed doctor profiles enable patients and administrative staff to easily identify and learn more about each doctor, facilitating better communication and coordination.
4. **Efficient Appointment Management System:**
   * The appointment management system allows doctors to schedule, reschedule, and manage appointments with patients seamlessly. It provides an intuitive interface for viewing available time slots, selecting appointment types (e.g., in-person, virtual), and confirming appointments. Doctors can also receive alerts and reminders for upcoming appointments, ensuring timely attendance and reducing no-show rates.
5. **Integration with Patient Details and Medical History:**
   * This feature integrates patient details and medical history within the Doctor Management Module, allowing doctors to access relevant information during patient consultations. It retrieves data from the patient management module or electronic health records (EHR) system and displays it alongside appointment details. Medical history may include past diagnoses, treatments, medications, allergies, and test results, enabling doctors to make informed decisions and provide personalized care.
6. **User-Friendly Frontend Website for Customers:**
   * The frontend website serves as a user-friendly platform for patients to interact with the healthcare facility. It allows patients to schedule appointments, view doctor profiles, access healthcare resources, and communicate with doctors or administrative staff. The website features intuitive navigation, clear layout, and responsive design to enhance user experience and accessibility across different devices.

#### Business Objectives:

1. **Increase Patient Satisfaction and Loyalty:** By providing personalized and efficient healthcare services, ensuring seamless access to appointments, and fostering trust and loyalty among patients.
2. **Improve Resource Utilization and Staff Productivity:** By streamlining processes, automating administrative tasks, and optimizing staff schedules to maximize productivity and efficiency.
3. **Enhance the Reputation and Competitiveness of the Healthcare Facility:** By offering advanced, user-friendly services to patients and doctors, positioning the facility as a leader in innovative, patient-centric care, and attracting patients and top talent alike.
4. **Reduce Operational Costs**: By streamlining administrative processes and reducing manual work, the module aims to lower operational costs associated with staffing, paperwork, and inefficiencies.
5. **Increase Revenue Generation:** By improving patient satisfaction and loyalty, the module can lead to increased patient retention and referrals, ultimately contributing to higher revenue generation for the healthcare facility.
6. **Facilitate Data-Driven Decision Making**: By centralizing patient data and appointment information, the module enables healthcare facilities to gather insights and analytics, facilitating data-driven decision-making to improve operational efficiency and patient care outcomes.

#### User Roles:

#### The Doctor Management Module caters to the following user roles:

#### Doctors: Responsible for managing appointments, accessing patient information, and updating their profiles.

#### Patients: Utilize the frontend website to schedule appointments, view doctor profiles, and access relevant healthcare information.

#### Administrators: Oversee the overall operation of the module, manage user accounts, and ensure system integrity and security.

#### Business Requirements:

1. User Authentication and Authorization:
   * The system must ensure secure login for doctors, patients, and administrative staff, with role-based access control to manage permissions effectively.
2. Appointment Scheduling and Management:
   * The module should provide an intuitive interface for scheduling, rescheduling, and canceling appointments, with features for managing appointment types, availability, and notifications.
3. Patient Information Management:
   * The system should allow for the efficient management of patient information, including demographics, medical history, insurance details, and consent forms, ensuring accuracy, privacy, and compliance with regulatory requirements.
4. Doctor Profile Management:
   * Doctors should have the ability to update their profiles with accurate and up-to-date information, including qualifications, specialties, availability, and contact details, to ensure transparency and trust with patients.
5. Appointment Reminders and Notifications:
   * The system should provide automated appointment reminders and notifications to patients via email, SMS, or mobile app notifications, reducing no-show rates and improving appointment adherence.
6. Scalability and Flexibility:
   * The module should be scalable and flexible to accommodate future growth, supporting an increasing number of doctors, patients, appointments, and system users without compromising performance or usability.

* **Frontend Website for Customers**

Frontend Website for Customers: The frontend website for customers provides patients with a user-friendly platform to:

* Schedule appointments with doctors based on availability and specialization
* View detailed profiles of doctors, including qualifications, experience, and availability
* Access relevant healthcare information, including tips, articles, and resources

#### Constraints:

## Despite its robust functionality, the Doctor Management Module operates within certain constraints, including:

## Budgetary constraints: Limited resources may impact the scope and scale of the module's development and implementation.

## Technical constraints: Compatibility issues with existing systems or technologies may pose challenges during integration and deployment.

## Regulatory constraints: Compliance with healthcare regulations and data privacy laws may impose limitations on data handling and system functionality.

**Technical Details:**

**Frontend:**

* **HTML**: Used for structuring the content of web pages, ensuring semantic markup and accessibility.
* **CSS**: Utilized for styling the user interface elements, adhering to modern design principles and providing a consistent visual experience.
* **JavaScript**: Employed to enhance the interactivity and functionality of the user interface, facilitating dynamic content updates and asynchronous communication with the server.

**Backend (Node.js):**

* **JavaScript**: Leveraged as the primary programming language for the backend, utilizing Node.js runtime environment for server-side development.

**Database (MS SQL Server):**

* **MS SQL Server**: Chosen as the relational database management system (RDBMS) for storing and managing data related to doctors, patients, appointments, and medical records.
* **SQL (Structured Query Language)**: Utilized for querying, manipulating, and managing data stored in the MS SQL Server database, enabling efficient data retrieval and manipulation operations.

**Integration:**

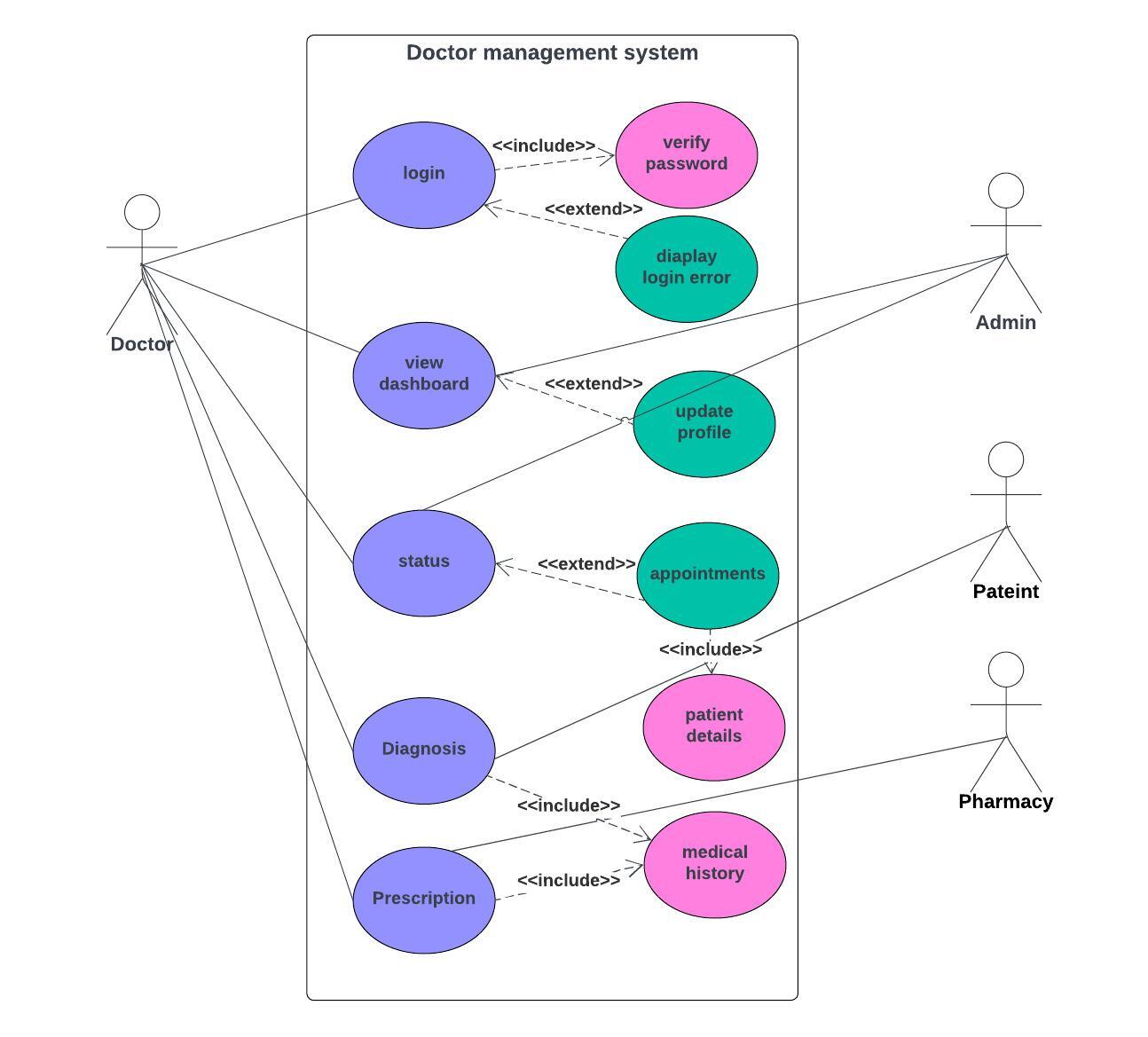
* **Backend-Database Integration**: Achieved using database drivers and modules compatible with MS SQL Server in the Node.js environment, facilitating seamless communication and data exchange between the backend application and the database.
* **Frontend-Backend Communication**: Implemented using HTTP protocols and RESTful APIs, enabling communication between the frontend user interface and the backend server for data retrieval, submission, and processing.

**API Endpoints**

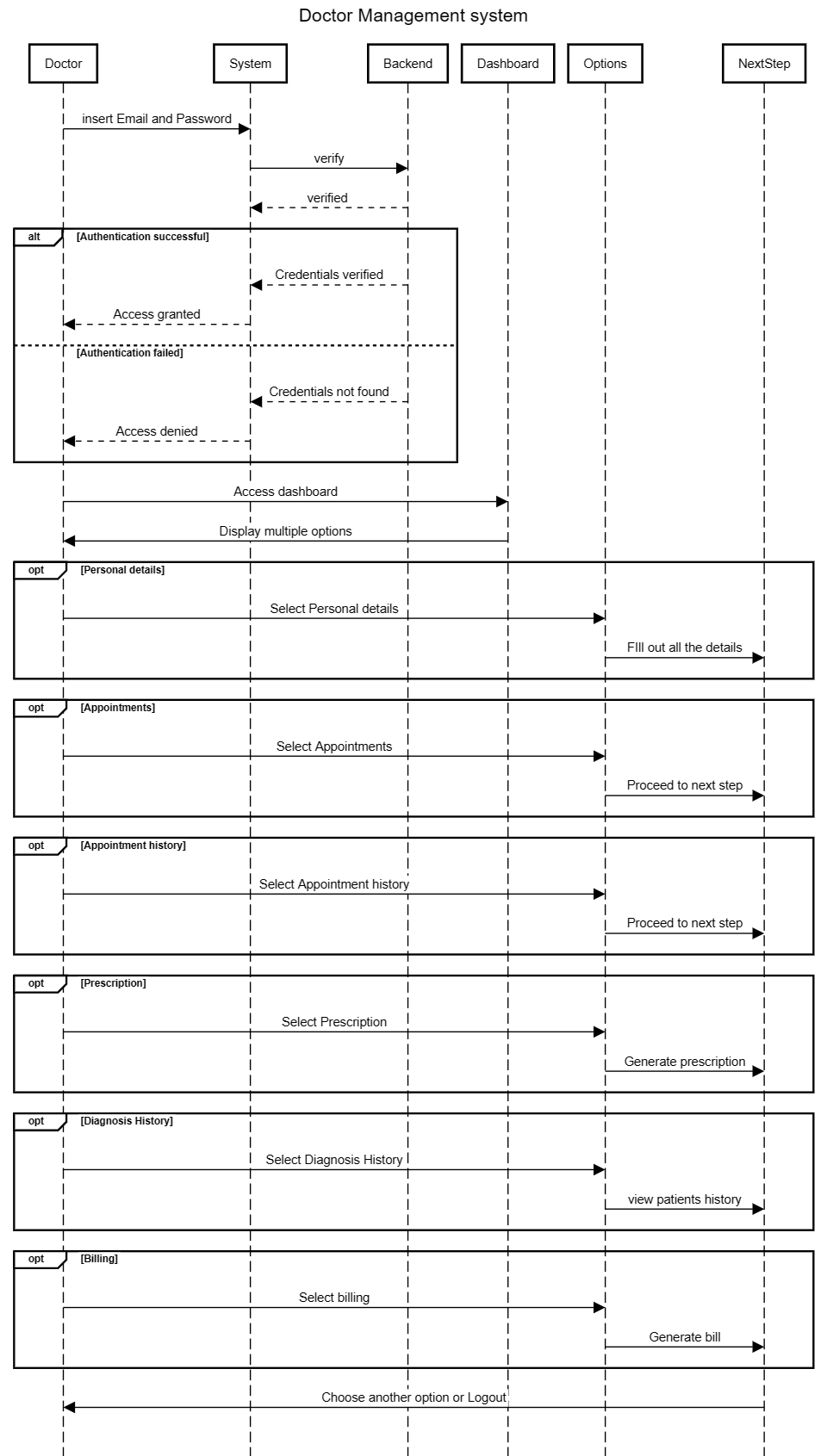
* **Login**: **/api/login**
  + Description: Authenticates doctor credentials.
  + Method: POST
* **Get Dashboard Data**: **/api/dashboard**
  + Description: Fetches all relevant dashboard information.
  + Method: GET
* **Update Personal Information**: **/api/doctor/update**
  + Description: Updates doctor's personal information.
  + Method: PUT
* **Appointments**:
  + **GET Appointments**: **/api/appointments**
    - Description: Retrieves all appointments.
    - Method: GET
  + **Generate Prescription**: **/api/appointments/{id}/prescription**
    - Description: Generates a prescription for a specific appointment.
    - Method: POST
* **Lab Tests**:
  + **Retrieve Lab Tests**: **/api/labtests**
    - Description: Retrieves lab tests based on appointment and test ID.
    - Method: GET
* **Diagnosis History**:
  + **Retrieve Diagnosis History**: **/api/diagnosis**
    - Description: Retrieves diagnosis history of patients.
    - Method: GET

**Diagrams**

**Use Case:**



**Sequence Diagram:**

****

## CHAPTER 01: UI / UX

### 1.1 Requirements

This UI/UX Team Requirements Document outlines the specific guidelines and expectations for the design aspects of our Doctor Management Module. The aim is to create an aesthetically pleasing, user-friendly, and immersive interface that enhances the overall user experience.

This UI/UX Team Requirements Document serves as a guide for the design team, providing a structured framework to deliver a visually stunning and user-centric interface.

#### 1.1.1 Design Principles

**- Clarity and Simplicity:** Ensure that the design is clear, simple, and intuitive, catering to users with varying levels of technological proficiency.

**- Consistency:** Maintain a consistent design language throughout the application to create a cohesive user experience.

**- Accessibility:** Prioritize accessibility features to make the application usable for individuals with diverse needs and abilities.

#### 1.1.2 Color Palette

- **Warm and Inviting Colors:** Choose colors that evoke a sense of comfort and warmth, aligning with the positive emotions associated with food.

- **Contrast and Readability:** Ensure high contrast between text and background colors for optimal readability.

- **Accents for Engagement:** Use vibrant accent colors strategically to draw attention to key elements and enhance user engagement.

#### 1.1.3 Typography

- **Readability:** Select fonts that prioritize readability on various devices and screen sizes.

- **Consistent Typography:** Maintain consistency in font choices across the application to establish a unified visual identity.

- **Hierarchy and Emphasis:** Utilize font sizes and weights to establish a clear hierarchy and emphasize important information.

#### 1.1.4 Web UI

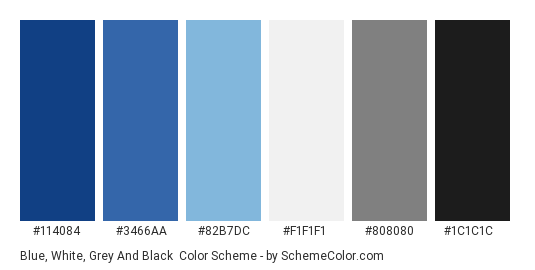
- **Responsive Layouts:** Design layouts that respond effectively to different screen sizes and resolutions on web browsers.

- **Cross-Browser Compatibility:** Ensure compatibility with major web browsers to guarantee a consistent user experience.

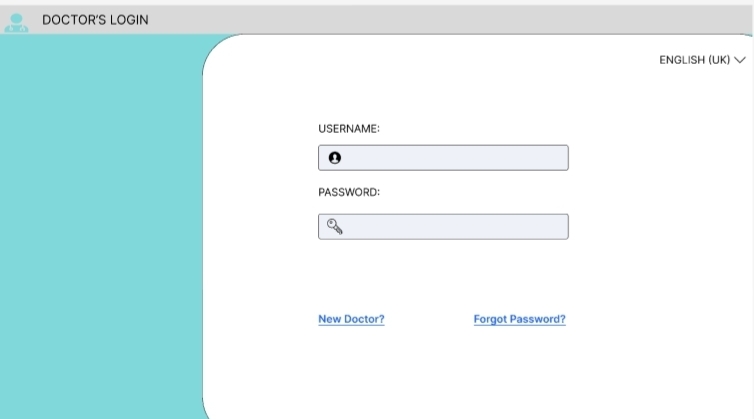
- **Navigation Efficiency:** Implement intuitive navigation for a seamless browsing experience.

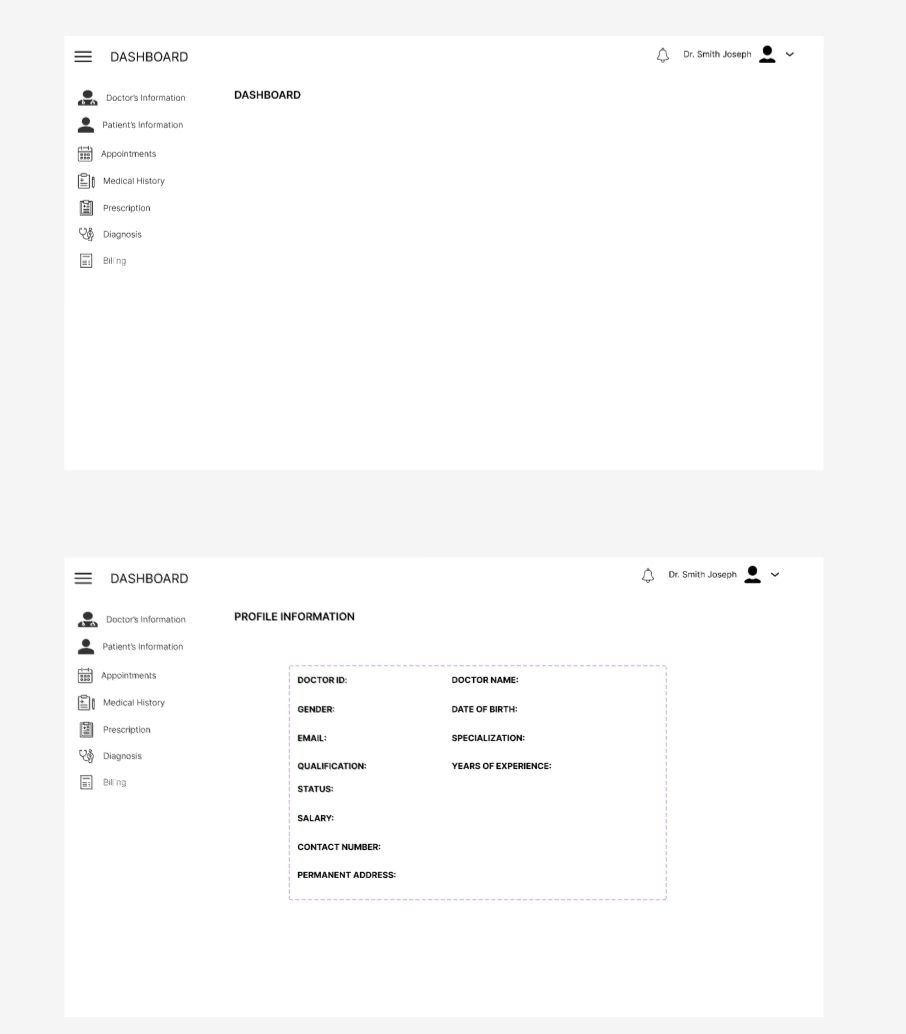
### 1.2 Color Palette

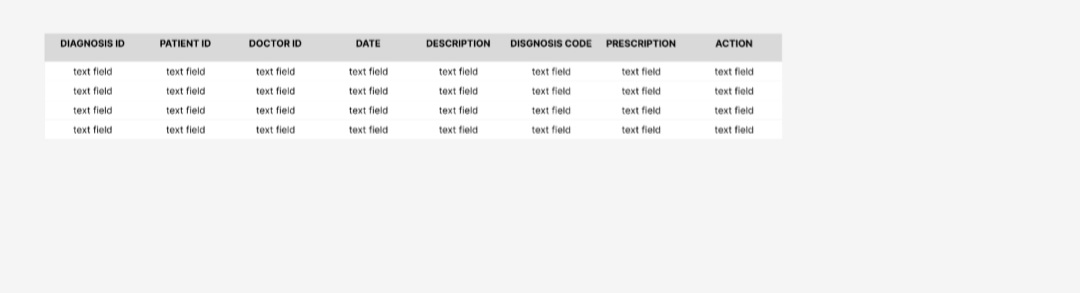
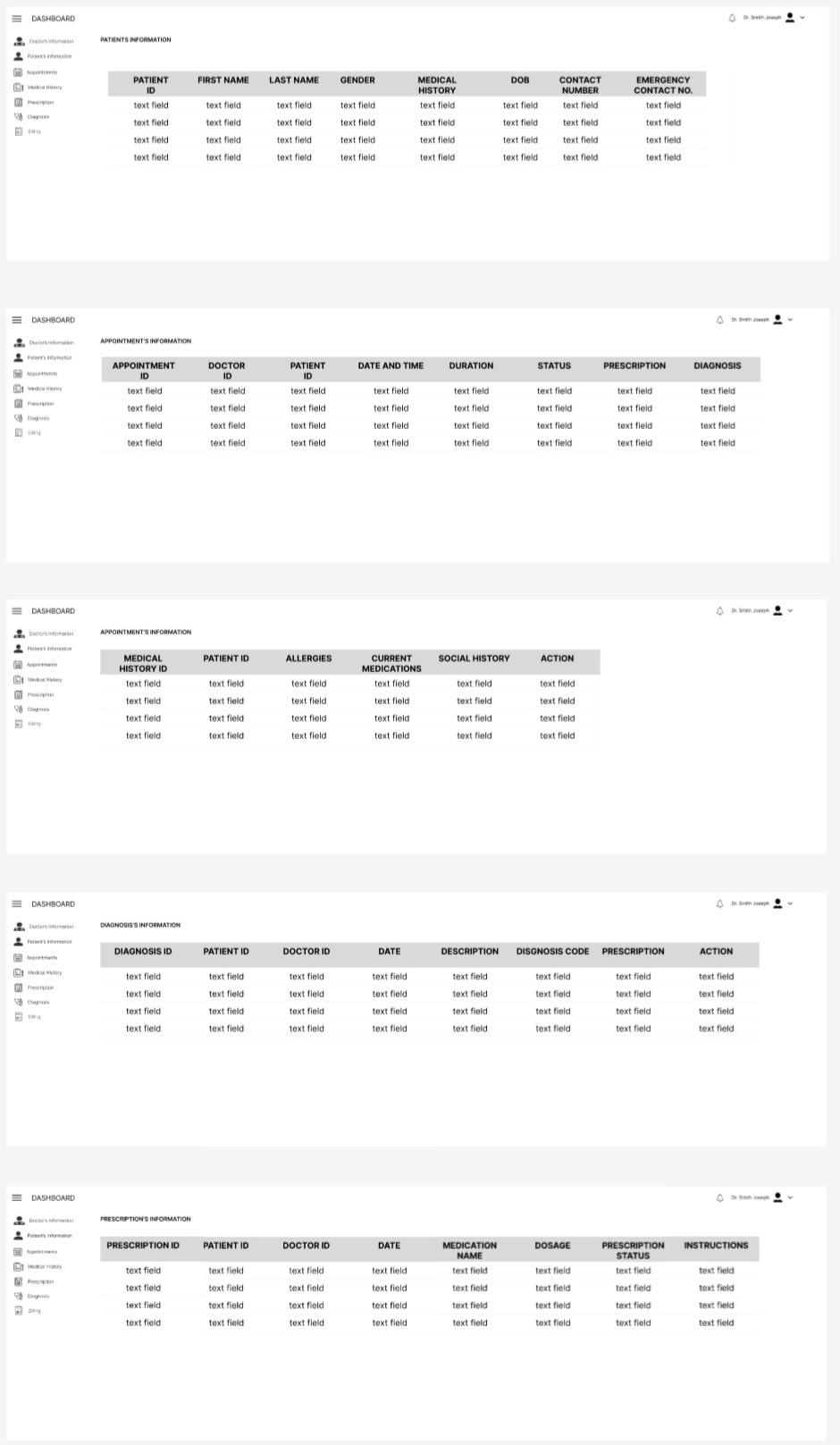
#### 1.2.1 Main Palette



### 1.3 Wireframes and Mockups







## 

## CHAPTER 02: DATABASE

### 2.1 Introduction

The database serves as the backbone of the Doctor Management Module, playing a crucial role in storing, managing, and retrieving data related to doctors, patients, appointments, and medical records. It serves as a centralized repository that facilitates efficient data management and supports the functionality of the module. The database design must be robust, scalable, and optimized to ensure data integrity, security, and performance.

### 2.2 Requirements

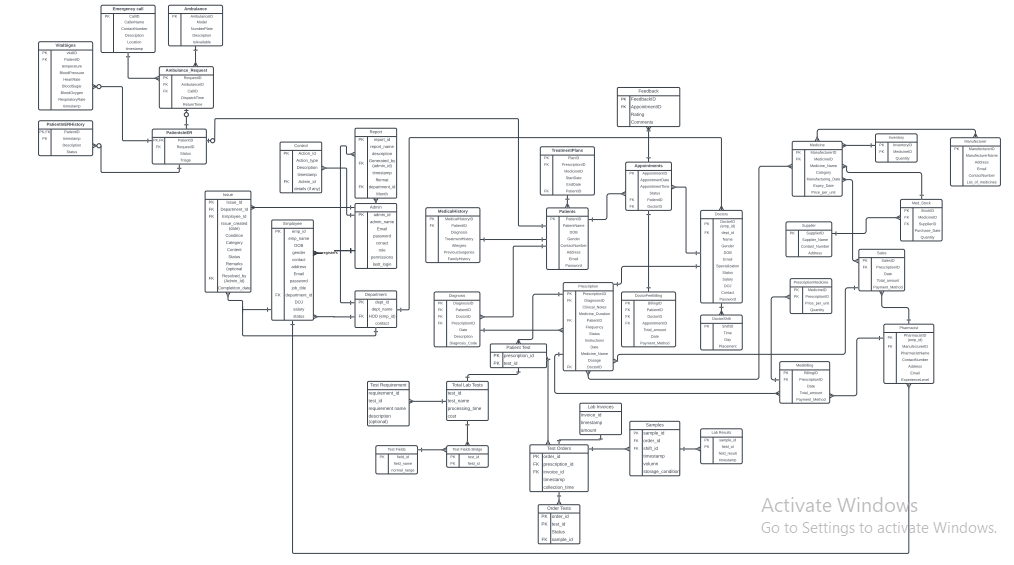
1. **Data Entities:**
   * Define the entities or tables within the database, including doctors, patients, appointments, medical records, and administrative staff, to represent different data entities within the healthcare facility.
2. **Attributes and Data Types:**
   * Specify the attributes or fields associated with each data entity, along with their corresponding data types, lengths, and constraints, to ensure consistency and accuracy of data storage.
3. **Relationships and Cardinality:**
   * Establish relationships between different data entities, such as one-to-one, one-to-many, or many-to-many relationships, to represent how data entities are interconnected and related to each other within the database.
4. **Normalization:**
   * Apply normalization techniques to eliminate data redundancy and ensure data integrity, adhering to standard normalization forms (e.g., First Normal Form, Second Normal Form, Third Normal Form).
5. **Indexing and Query Optimization:**
   * Implement indexing strategies to improve query performance and optimize database operations, ensuring fast retrieval of data and efficient query execution.
6. **Security and Access Control:**
   * Implement security measures to protect sensitive data and ensure access control based on user roles and permissions, safeguarding against unauthorized access and data breaches.
7. **Scalability and Performance:**
   * Design the database architecture to be scalable and capable of handling increasing data volumes and user loads over time, while maintaining optimal performance and response times.
8. **Data Integrity and Constraints:**
   * Enforce data integrity constraints, such as unique constraints, foreign key constraints, and check constraints, to maintain data consistency and prevent data anomalies or inconsistencies.

### 2.3 ERD Description

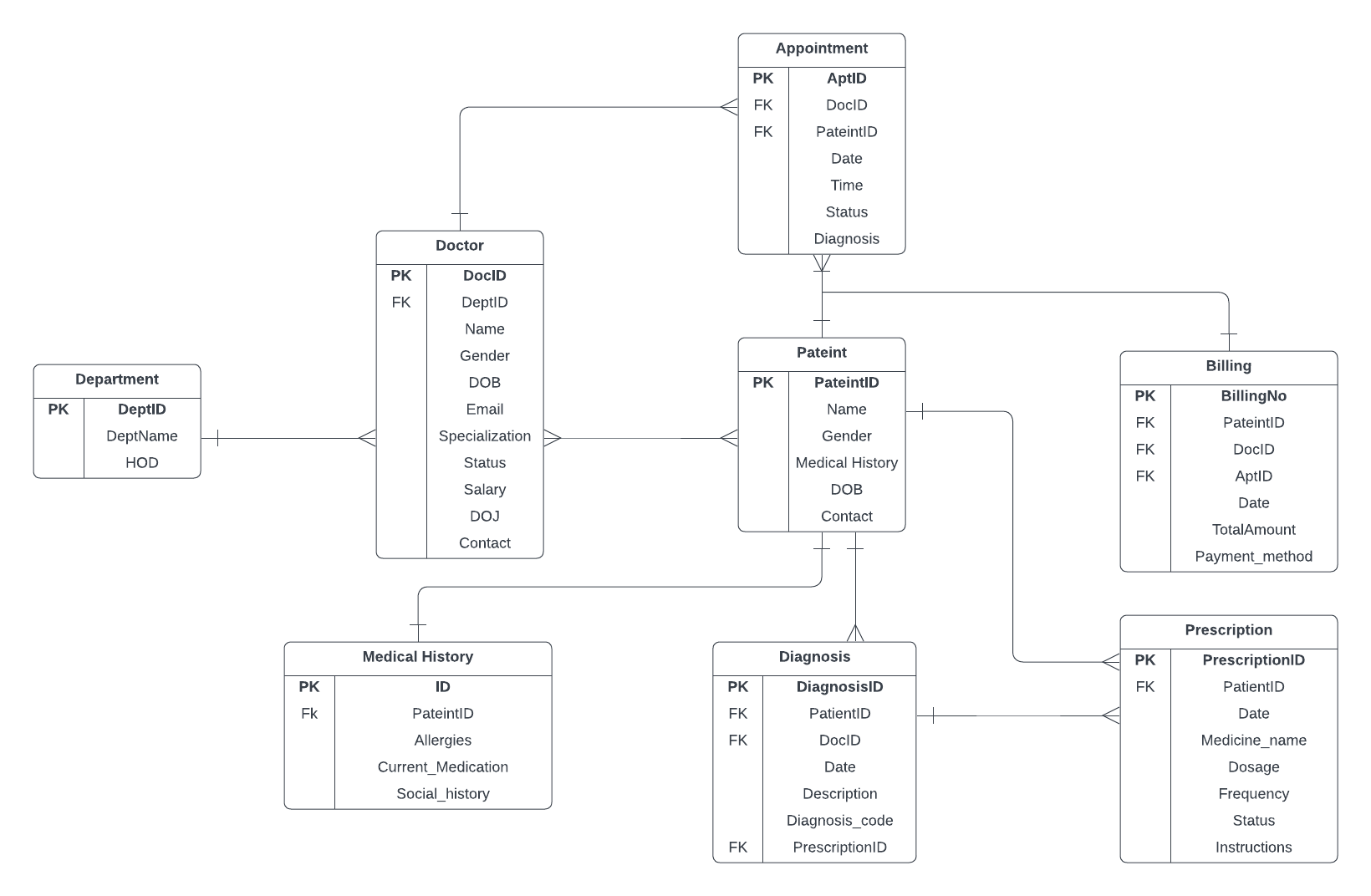
1. Each doctor can be associated with only one department, but a department can have multiple doctors.
2. A doctor can have many patients.
3. A patient can visit multiple doctors.
4. A doctor can have multiple appointments, but an appointment is associated with only one doctor.
5. A patient can have multiple appointments, but an appointment is associated with only one patient.
6. A patient can have one or more diagnoses
7. Each diagnosis is associated with a specific patient.
8. A diagnosis may result in one or more prescriptions.
9. Each prescription is related to a specific diagnosis.
10. A patient can have multiple prescriptions over time, but each prescription is associated with exactly one patient.
11. Each appointment leads to a billing event and each billing event is associated with exactly one appointment.
12. Each patient has a unique medical history that pertains specifically to them.
13. Each medical history record is associated with exactly one patient.

### 2.4 Diagrams

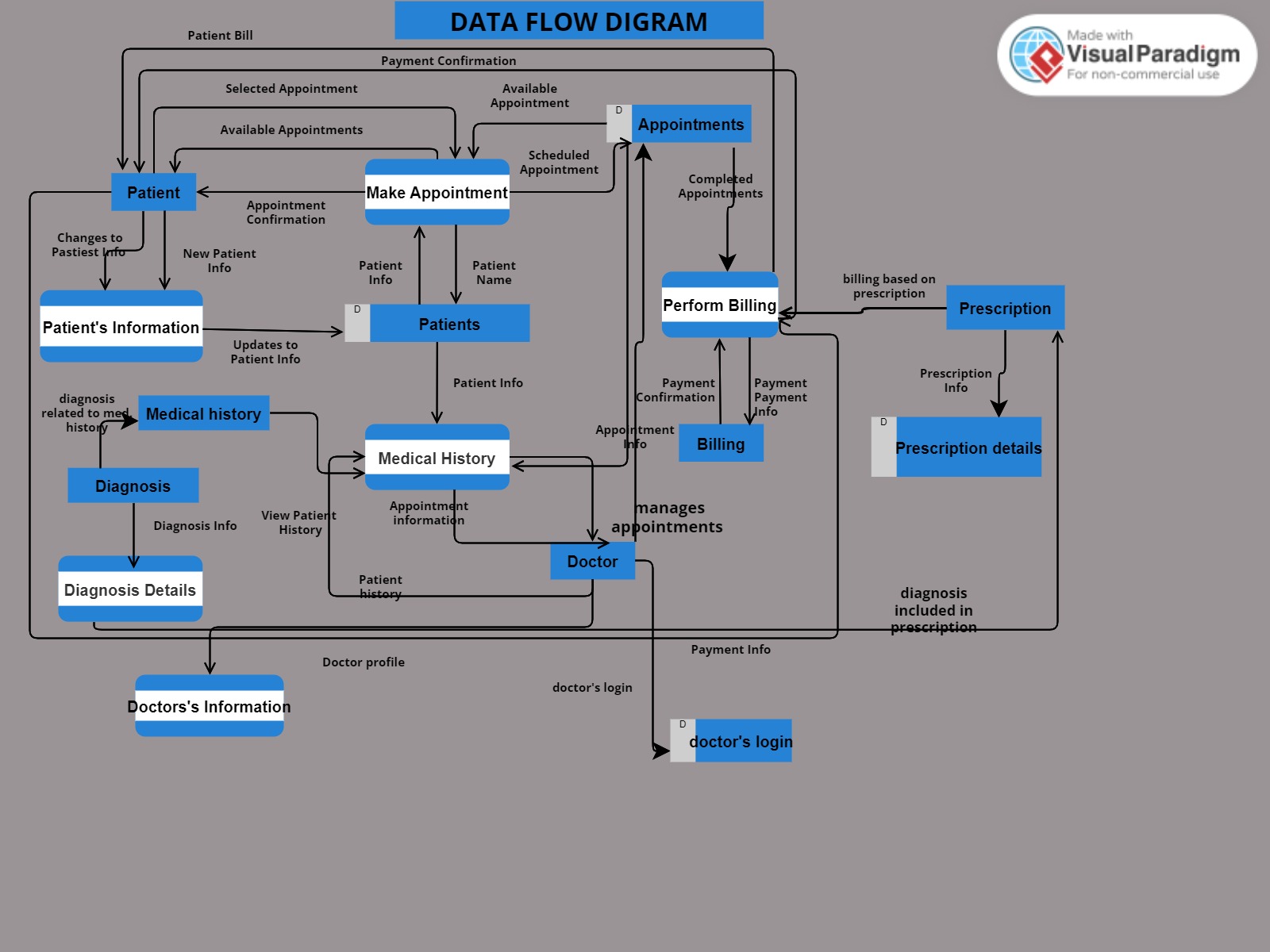
#### 2.4.1 Unified ERD Diagrams



**2.4.2 ERD**



#### 2.4.3 DFD Level 1 Diagram



### 2.5 Entities/Attributes

|  |  |
| --- | --- |
| **ENTITIES** | **ATTRIBUTES** |
| Doctor | DoctorID, First Name, Last Name, Gender, DOB, Email, Specialization, Status, Years of experience, Salary, DOJ, Contact number |
| Department | DeptID, Dept Name, HOD |
| Patient | PatientID, First Name, Last Name, Gender, Medical History, DOB, Contact #, emergency contact number, |
| Appointment | AppointmentID, DoctorID, PatientID, Date & Time, Duration, Status, Prescription, Diagnosis |
| Diagnosis | DiagnosisID, PatientID, DoctorID, Date, Description, Diagnosis Code, Prescriptions |
| Prescription | PrescriptionID, DoctorID, PatientID, Prescription Date, Medication Name, Dosage, Frequency, Prescription Status, Instructions |
| Medical History | MHistory ID, PatientID, Allergies, Current medication , Social History |
| Billing | BillingID, PatientID, DoctorID, AppointmentID, Date, Total Amount, Payment Method. |

## CHAPTER 03: Web Application

### 3.1 Introduction

### The web application for the Doctor Management Module serves as a central platform for doctors, patients, and administrative staff to interact and manage various aspects of healthcare services. It provides a user-friendly interface accessible via web browsers, offering functionalities such as appointment scheduling, patient information management, and communication tools. The design and development of the web application are crucial to ensure seamless user experience, data security, and efficient workflow within the healthcare facility.

### 3.2 Requirements

1. **Functional Requirements:**
   * Define the core functionalities of the web application, including appointment management, patient information access, doctor profile updates, communication features, and administrative tools.
2. **User Interface Design:**
   * Specify the design requirements for the user interface, including layout, navigation, visual elements, and responsive design principles to ensure compatibility across different devices and screen sizes.
3. **Backend Integration:**
   * Outline the integration requirements with backend systems, such as the database management system and external APIs, to facilitate data exchange and system interoperability.
4. **Security Measures:**
   * Identify security requirements, such as data encryption, secure authentication mechanisms, access control, and protection against common web application vulnerabilities (e.g., SQL injection, cross-site scripting), to safeguard sensitive information and prevent unauthorized access.
5. **Performance Optimization:**
   * Define performance requirements, including page load times, server response times, and scalability considerations, to ensure optimal performance and responsiveness of the web application, even under high traffic loads.
6. **Compatibility and Browser Support:**
   * Specify compatibility requirements for web browsers and devices, ensuring that the web application functions correctly across popular browsers (e.g., Chrome, Firefox, Safari) and operating systems (e.g., Windows, macOS, iOS, Android).

### 3.3 Functional Requirements

**Doctor Login Panel**

- The system shall provide a secure login interface for doctors.

- Doctors shall be required to enter their credentials (username and password) for authentication.

- Authentication shall be performed securely to prevent unauthorized access.

**Doctor Dashboard**

- Upon successful login, doctors shall be redirected to a personalized dashboard.

- The dashboard shall provide an overview of upcoming appointments, recent patient interactions, and other relevant notifications.

- Doctors shall have access to quick links for managing appointments and updating their profile.

**Doctor Profile**

- The system shall maintain a profile for each doctor.

- Doctor profiles shall include the following details:

- Name

- Contact Information

- Experience

- Department/Specialization

- Salary

- Qualifications

- Schedule/Availability

- Profile Picture

**Doctor Appointments**

- Doctors shall be able to view and manage their appointments through the system.

- The system shall allow doctors to schedule new appointments and modify existing ones.

- Appointment details shall include:

- Patient Name

- Appointment Date and Time

- Reason for Appointment

- Status (e.g., confirmed, pending)

- Appointment Type (e.g., in-person, virtual)

**Patient Details in Appointments**

- Within each appointment, doctors shall have access to patient details and medical history.

- Patient details shall be retrieved from the database and displayed alongside the appointment information.

- Medical history shall include relevant information such as past diagnoses, prescribed medications, allergies, and test results.

**3.4. Non-Functional Requirements**

**Security**

- The system shall implement robust security measures to safeguard doctor and patient data.

- Access to sensitive information shall be restricted based on user roles and permissions.

- Data transmission shall be encrypted to prevent interception by unauthorized parties.

**Performance**

- The system shall be capable of handling concurrent access by multiple doctors and patients.

- Response times for retrieving patient data and updating appointments shall be optimized to ensure a seamless user experience.

**Scalability**

- The system architecture shall be designed to accommodate future scalability requirements.

- It shall be capable of scaling up to support an increasing number of doctors, patients, and appointments without significant performance degradation.

**User Interface**

-The user interface shall be intuitive and user-friendly, facilitating easy navigation for doctors of varying technical proficiency.

-It shall adhere to modern design principles and accessibility standards to cater to diverse user demographics.

-The user interface shall be developed using HTML, CSS, and JavaScript to ensure compatibility across modern web browsers and devices.

-HTML shall be used for structuring the content of web pages, ensuring semantic markup and accessibility.

-CSS shall be utilized for styling the user interface elements, adhering to modern design principles and providing a consistent visual experience.

-JavaScript shall be employed to enhance the interactivity and functionality of the user interface, facilitating dynamic content updates and asynchronous communication with the server.

-The user interface design shall be responsive, adapting seamlessly to various screen sizes and resolutions to provide an optimal viewing experience on desktop and mobile devices.

### 3.4 Module Handling

#### 3.4.1 Integration with Other Modules

The Doctor Management Module is designed to seamlessly integrate with various modules within the healthcare ecosystem, including lab, prescriptions, emergency, admin, and patients. These integrations enable efficient data exchange, streamline workflows, and enhance patient care across different departments and stakeholders within the healthcare facility.

**Key Integration Points:**

1. **Lab Module Integration:** Facilitates the transmission of lab test requests and results between the Doctor Management Module and the Lab Module, allowing doctors to order tests, retrieve results, and make informed decisions.
2. **Prescriptions Module Integration:** Enables doctors to generate electronic prescriptions, access medication history, and communicate with pharmacies directly from the Doctor Management Module, ensuring safe and efficient medication management.
3. **Emergency Module Integration:** Provides real-time alerts and notifications regarding critical patient conditions, facilitates patient transfer, and ensures seamless coordination of care during emergency situations.
4. **Admin Module Integration:** Syncs user accounts, roles, and permissions, and leverages reporting and analytics capabilities to support decision-making and administrative tasks across the healthcare facility.
5. **Patients Module Integration:** Streamlines patient registration, engagement, and communication, allowing patients to access appointments, medical records, and educational resources through the Doctor Management Module.

# [Chapter 4: BACKEND](#_Chapter_4:_BACKEND)

**4.1 Overview**

The backend of the Doctor Management Module serves as the core engine that handles data processing, business logic, and communication between the frontend user interface and the underlying database. It comprises various components and functions implemented using JavaScript to ensure seamless operation and functionality of the web application. Here's a comprehensive overview of the backend components and their roles:

1. **Server-Side Logic:**
   * The backend server, implemented using JavaScript (ssms, vs code, Node.js), hosts the application logic and handles client requests and responses.
   * It serves as the intermediary between the frontend user interface and the database, processing incoming requests, executing business logic, and generating appropriate responses.
2. **Routing and Endpoint Management:**
   * Backend routing mechanisms define the URL paths and corresponding endpoints that handle specific types of client requests
   * Endpoint management ensures proper routing of requests to the corresponding backend functions for processing and execution.
3. **Authentication and Authorization:**
   * Authentication mechanisms verify the identity of users logging into the system, ensuring that only authorized users gain access to protected resources.
   * Authorization controls access to different features and functionalities of the Doctor Management Module based on user roles and permissions.
4. **Data Access and Database Interaction:**
   * Backend components interact with the SQL Server database to retrieve, manipulate, and store data related to doctors, patients, appointments, and medical records.
   * Data access logic involves executing SQL queries, transactions, and database operations to fetch, update, or delete records as required by the application.
5. **Business Logic and Functionality:**
   * Backend logic implements the core functionalities of the Doctor Management Module, including appointment scheduling, patient information management, user authentication, and integration with other modules.
   * It encapsulates business rules, validation logic, and algorithms necessary to process user requests, enforce system policies, and maintain data integrity.
6. **Integration with External Services and APIs:**
   * The backend may integrate with external services, APIs, or third-party systems (e.g., lab systems, prescription databases) to exchange data and extend the functionality of the Doctor Management Module.
   * Integration logic handles communication with external services, formats data according to API specifications, and manages authentication and authorization for secure data exchange.
7. **Error Handling and Logging:**
   * Backend components implement error handling mechanisms to gracefully handle exceptions, errors, and unexpected conditions that may arise during application execution.
   * Logging functionality captures relevant events, errors, and debugging information for monitoring, troubleshooting, and performance analysis purposes.
8. **Security Measures:**
   * Backend incorporates security measures such as data encryption, input validation, and protection against common web vulnerabilities (e.g., SQL injection, cross-site scripting) to safeguard sensitive information and prevent unauthorized access.

## [4.2 Function and Description](#_4.1_Function_and)

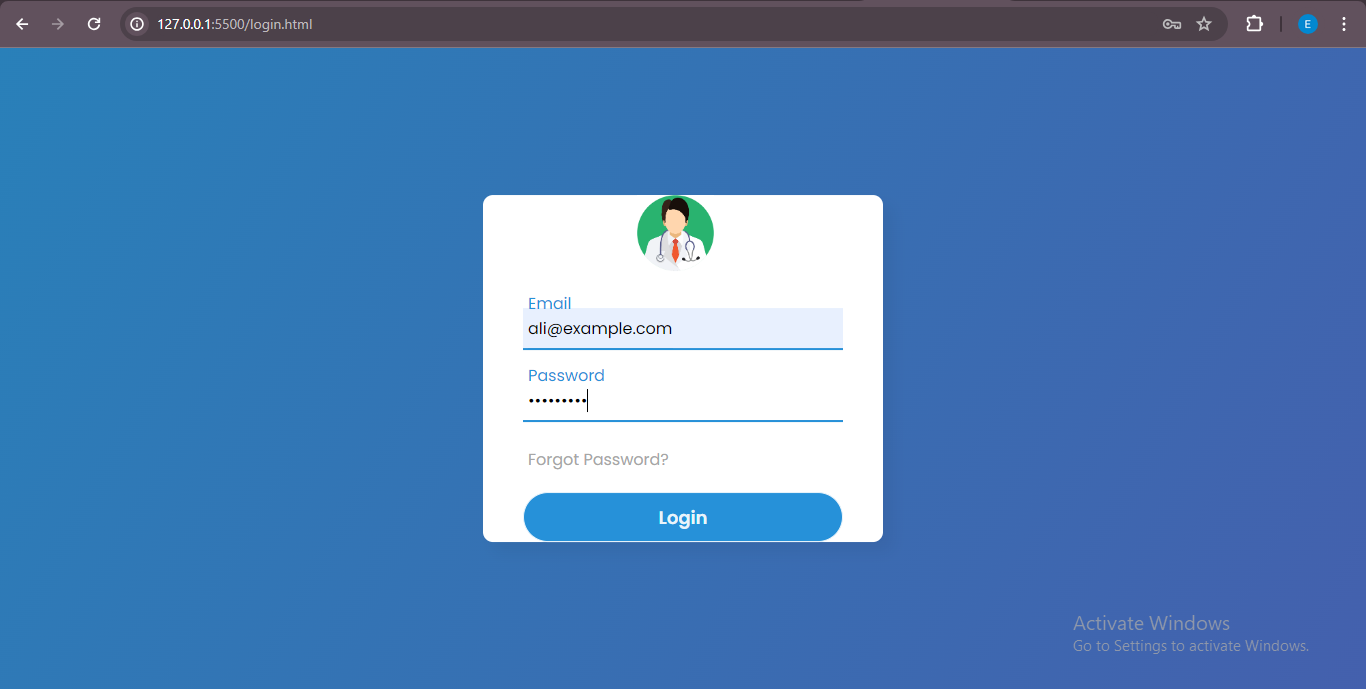
1. **User Login:**
   * Users input their credentials (username/email and password) on the login page of the web application.
   * JavaScript validates the input fields to ensure they are not empty and meet any specified format requirements.
   * Upon submission, the credentials are sent to the backend server for authentication.
2. **Authentication Process:**
   * The backend server receives the login request and queries the MySQL database to verify the provided credentials against stored user data.
   * JavaScript backend logic compares the hashed password stored in the database with the hashed password provided during login.
   * If the credentials match, the user is authenticated, and a session token is generated and returned to the client-side JavaScript.
3. **Querying Patient History:**
   * Upon authentication and authorization, the JavaScript backend retrieves patient history data from the MySQL database based on the patient's unique identifier (e.g., patient ID or medical record number).
   * The backend constructs SQL queries to fetch relevant information from the database tables storing patient records, diagnoses, treatments, medications, and allergies.
4. **Data Processing and Formatting:**
   * Retrieved patient history data is processed and formatted by the JavaScript backend to organize it in a structured and user-friendly manner.
   * The backend logic may perform data transformation, filtering, and sorting operations to present the information in a coherent and comprehensible format for the user.
5. **Presentation to User Interface:**
   * Processed patient history data is sent to the frontend user interface (HTML/CSS) for rendering and display to the user.
   * JavaScript backend may generate dynamic HTML content or JSON objects containing patient history details to be consumed by the frontend.
6. **User Interaction and Navigation:**
   * Users navigate through the patient history interface to view different sections, such as diagnoses, treatments, medications, and allergies.
   * JavaScript frontend provides interactive features, such as search filters, pagination, and sorting options, to facilitate efficient exploration of patient history data.
7. **Error Handling:**
   * JavaScript backend handles exceptions and errors gracefully, providing error messages or notifications to users in case of data retrieval failures or other issues.
8. **Querying Upcoming Appointments:**
   * Upon authentication and authorization, the JavaScript backend retrieves upcoming appointment data from the MySQL database.
   * The backend constructs SQL queries to fetch relevant information from the database tables storing appointment schedules, patient details, and appointment statuses.

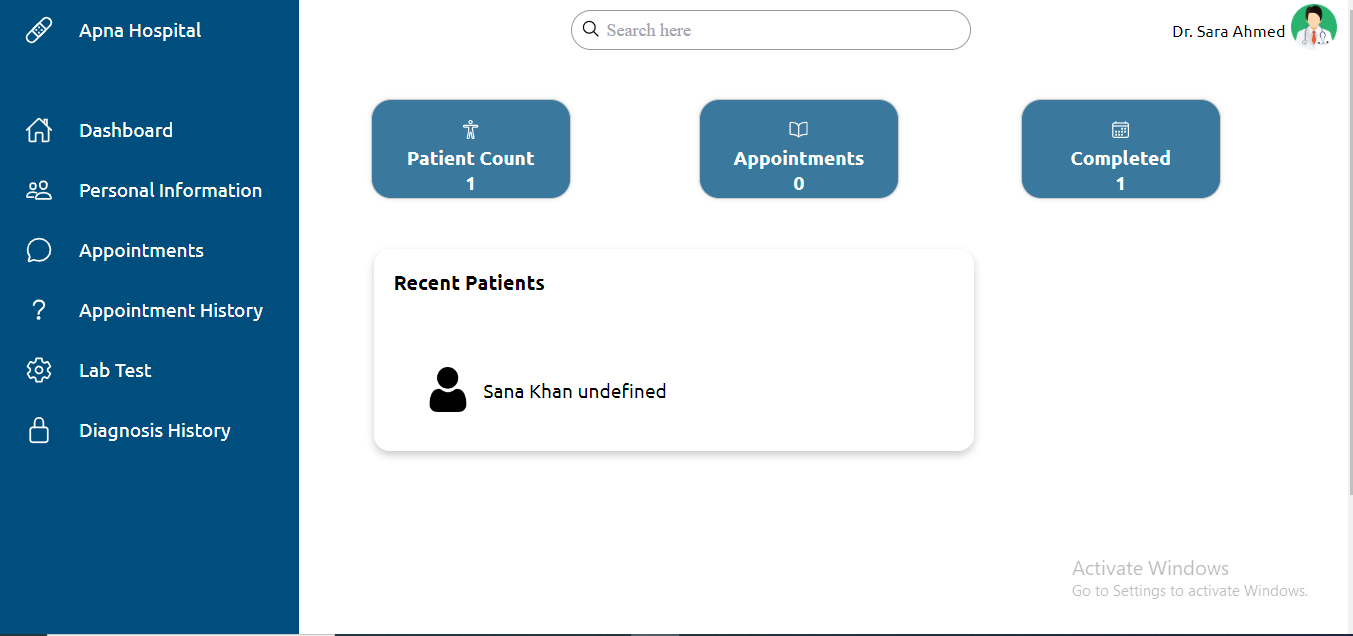
**Chapter 5: USER GUIDE**

**5.1 User Guide**

**Dashboard Overview**

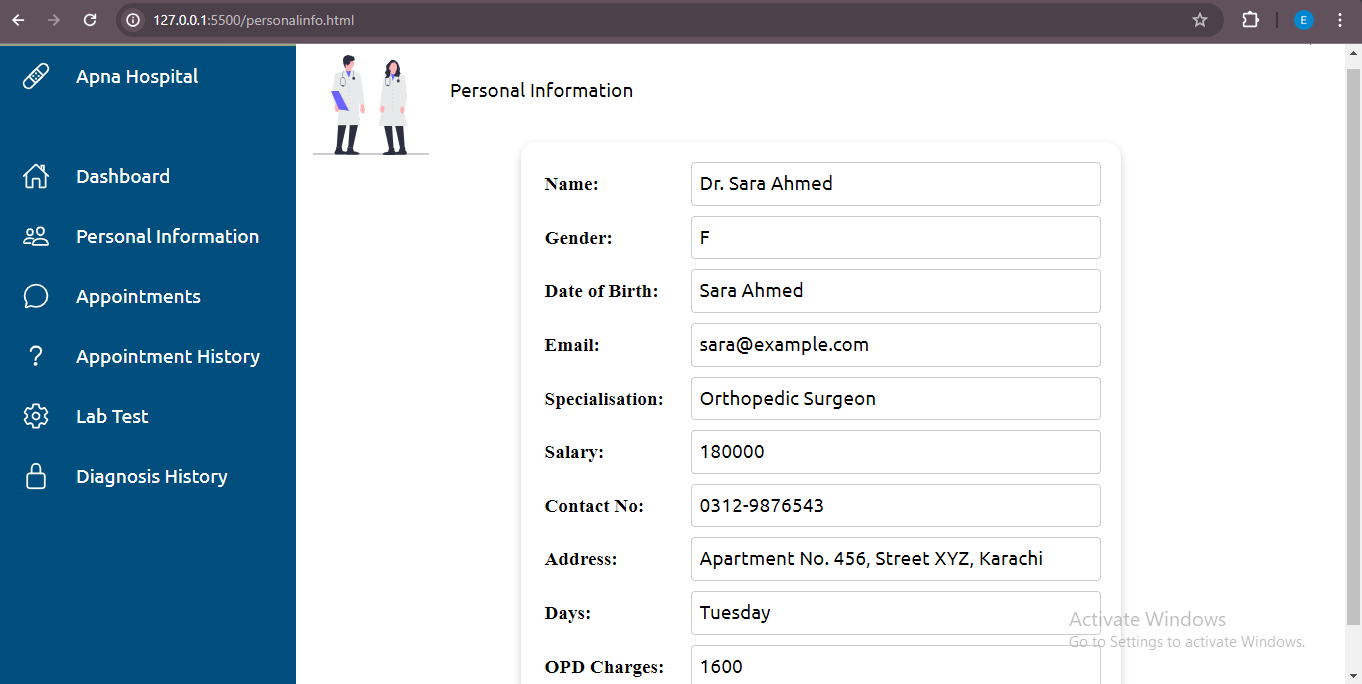
Upon logging in, doctors are greeted with a dashboard providing a comprehensive overview of their personal information, upcoming appointments, past appointment history, lab tests, and diagnosis history.





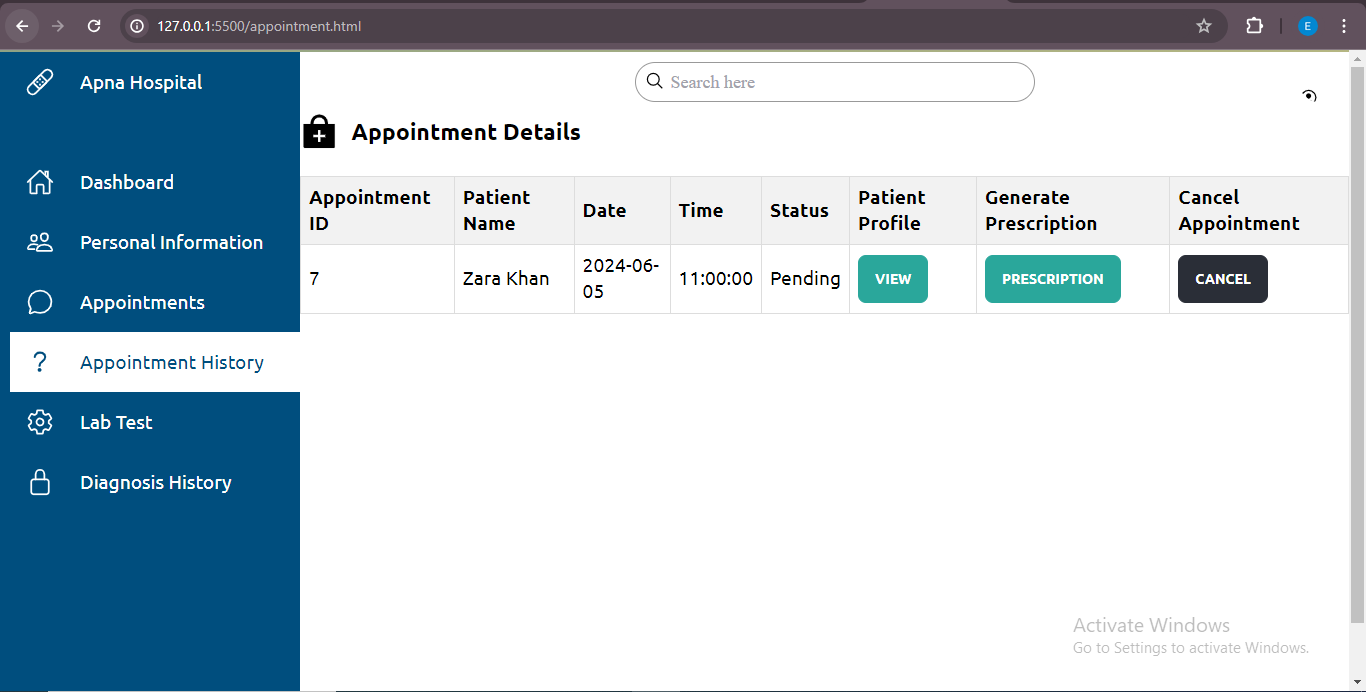
**Personal Information**

* **Details**: Displays essential information about the doctor, including their name, contact details, specialization, and other relevant details.

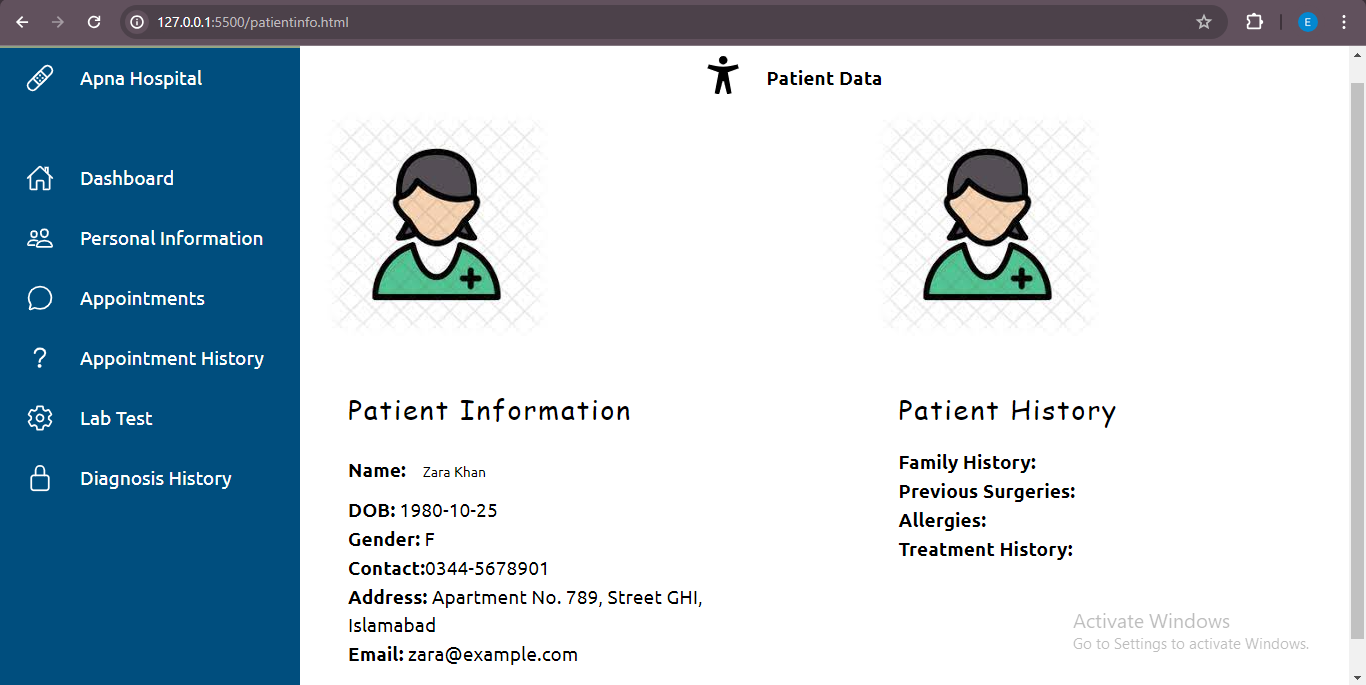


**Appointments**

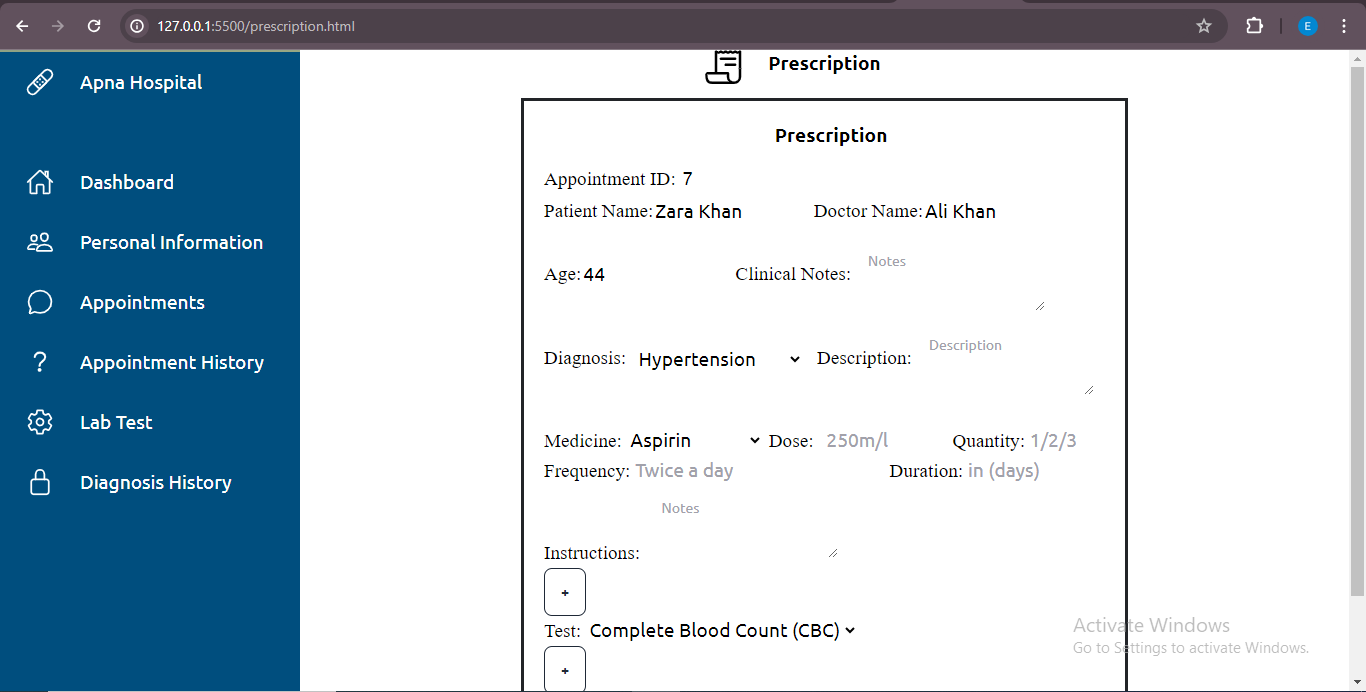
* **View Appointments**: Presents a list of upcoming appointments, including patient names and scheduled times.



* **Patient Profile**: Allows doctors to access the profile of each patient by clicking on their name within the appointment list.

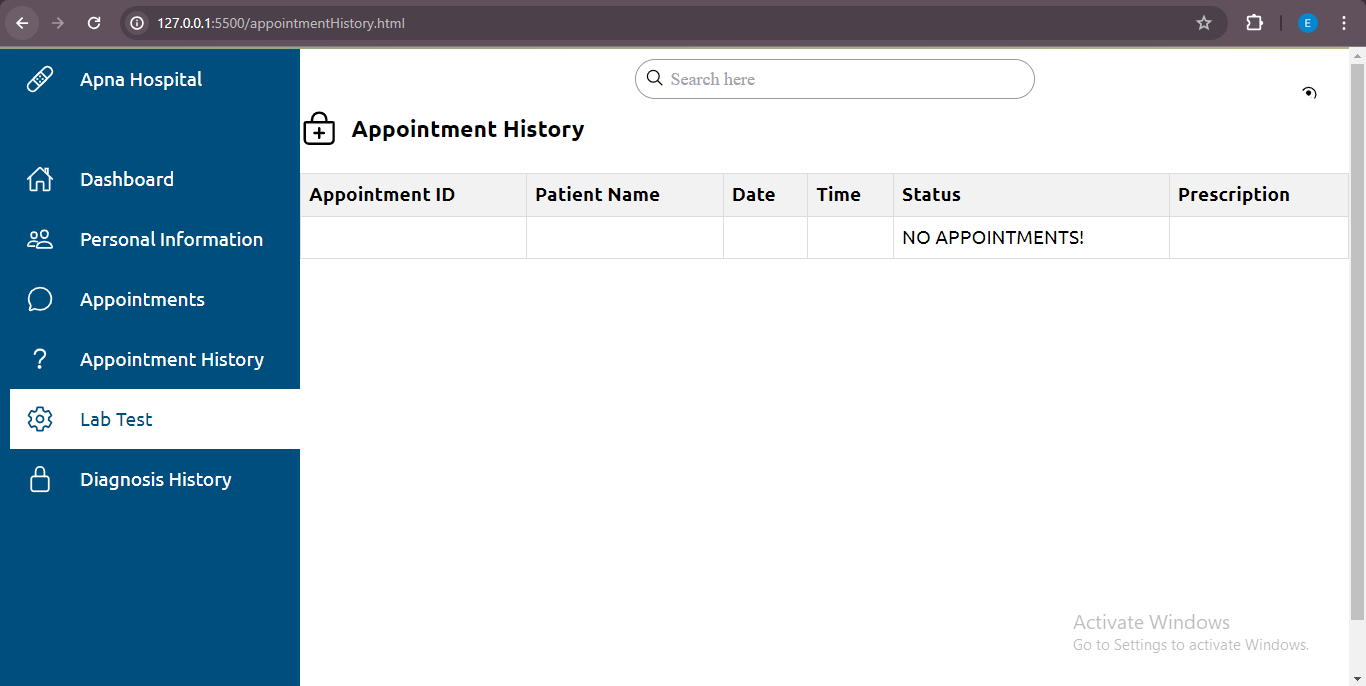


* **Generate Prescription**: Provides an option for doctors to create and save prescriptions for patients directly from the dashboard.



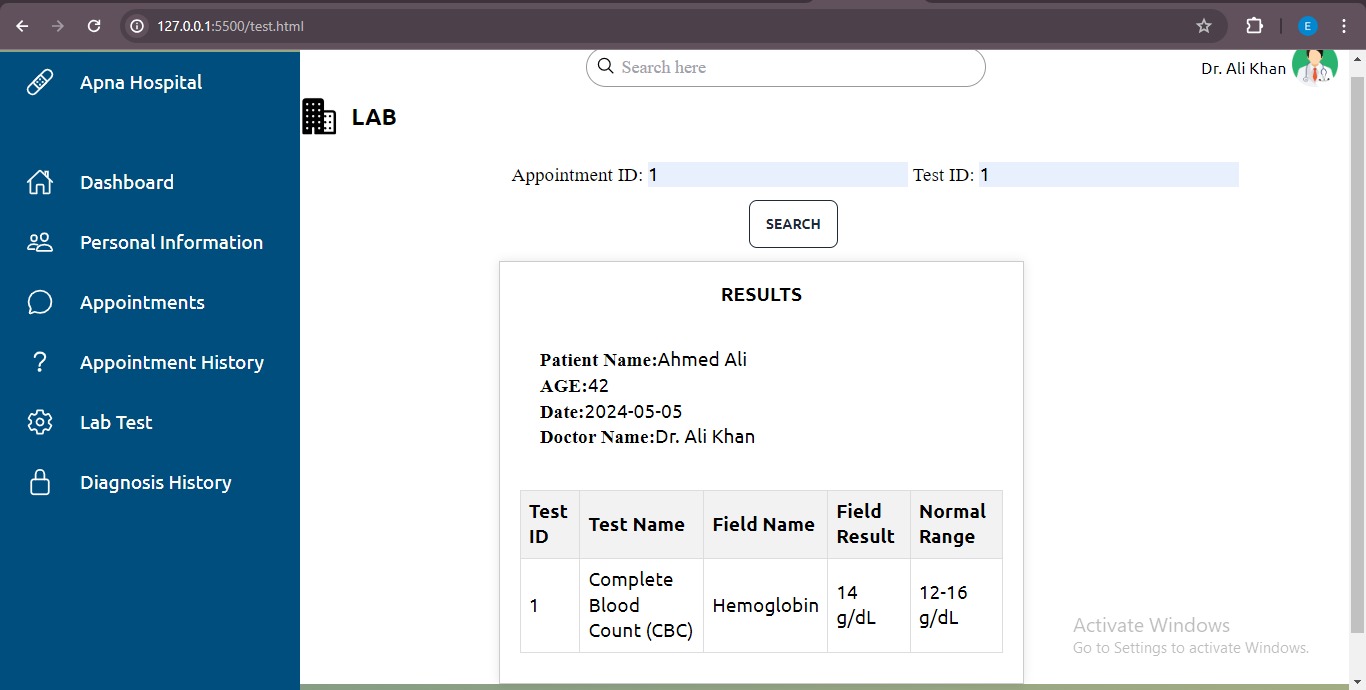
**Appointment History**

* **Past Patients**: Shows a list of patients previously seen by the doctor.
* **View Prescription**: Enables doctors to access prescriptions issued to past patients for reference or modification.



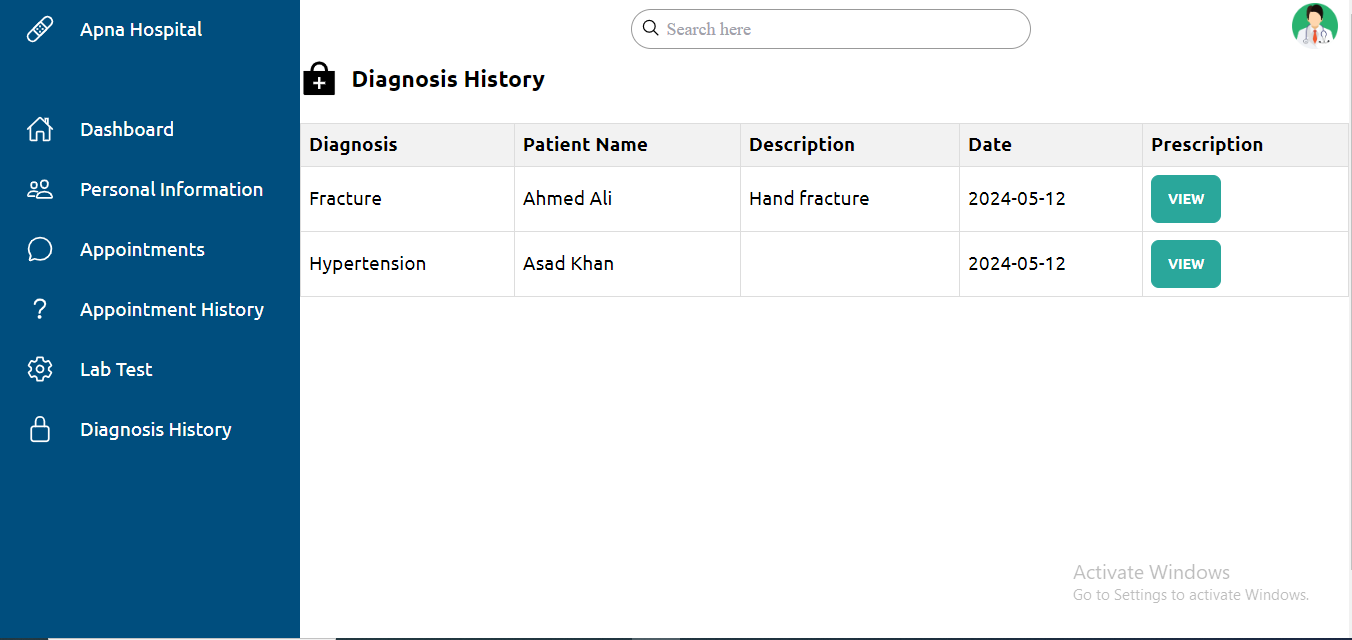
**Lab Tests**

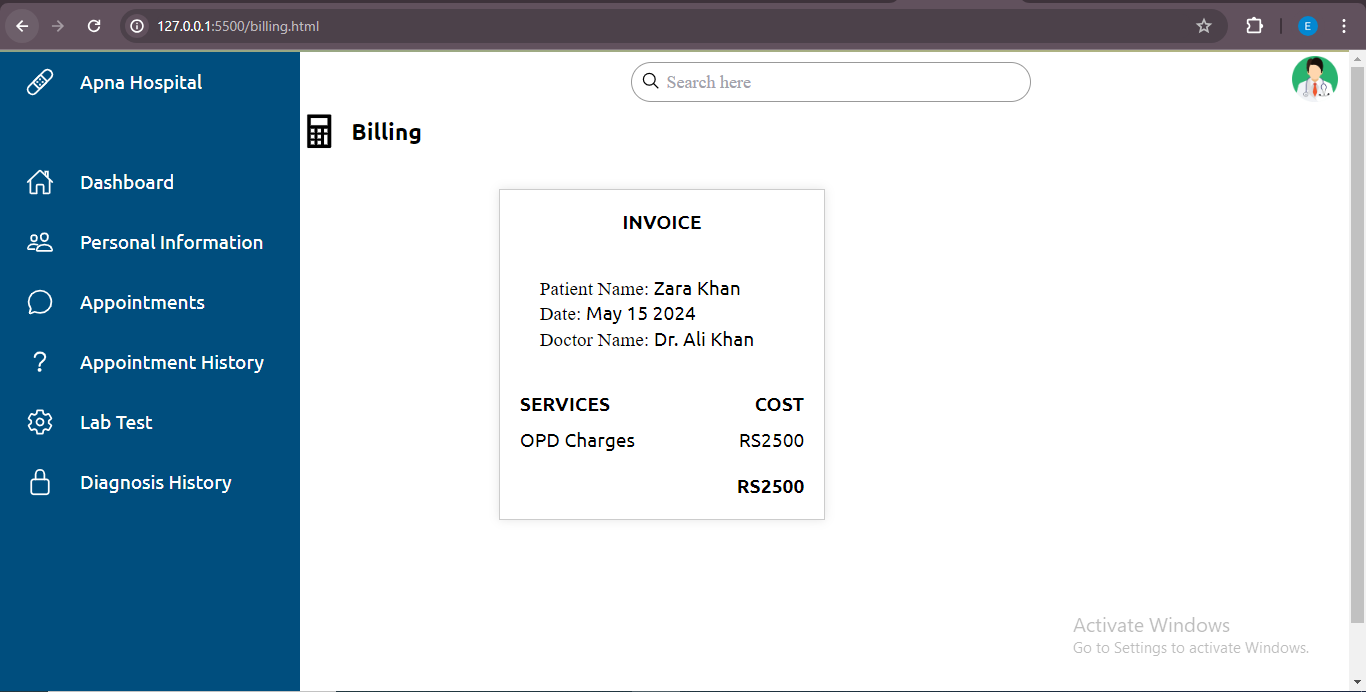
* **Enter Appointment and Test ID**: Allows doctors to input the appointment ID and test ID to retrieve relevant lab test information.
* **View Tests**: Displays all lab tests recommended by the doctor for the specified appointment, facilitating access to diagnostic information.



**Diagnosis History**

* **View Diagnosis**: Provides a comprehensive list of diagnoses made by the doctor for different patients.
* **View Prescription and Medical History**: Offers a detailed view of patient prescriptions and medical history, aiding doctors in making informed decisions during consultations.





**5.2 Testing:**

**Login Validation Testing:**

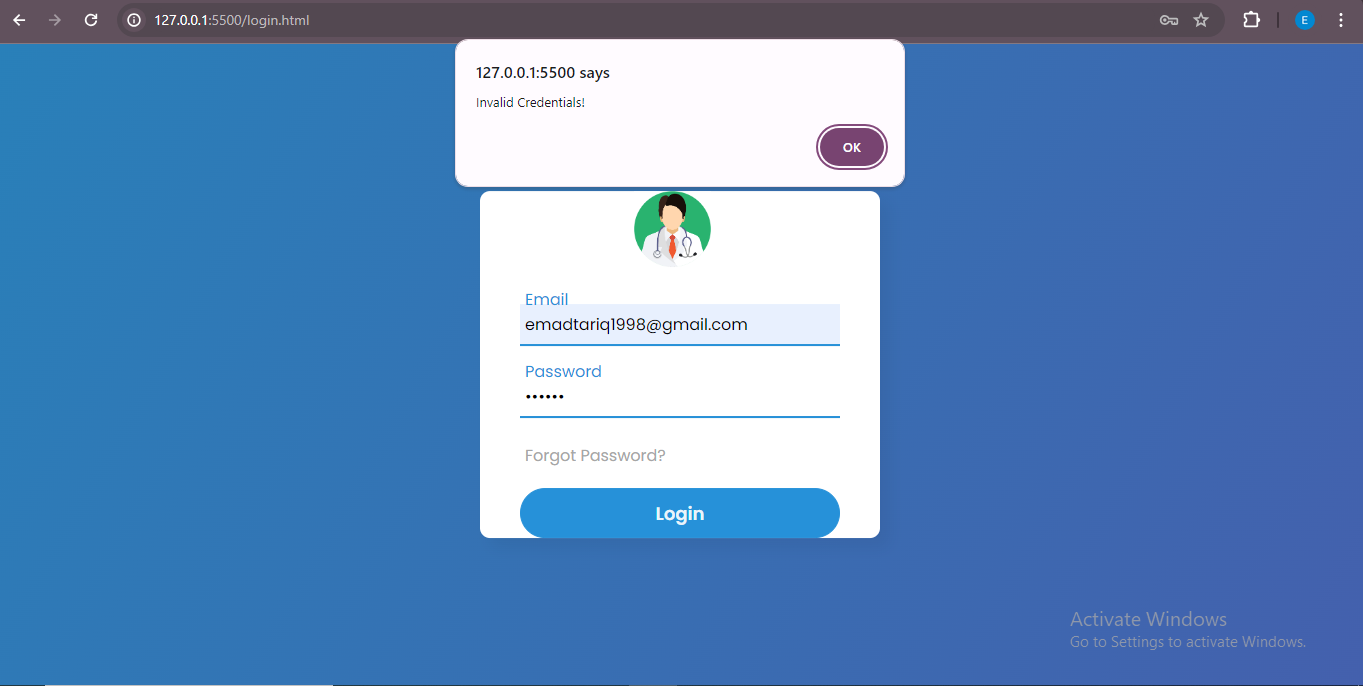
The login validation testing aimed to verify the system's behavior when users attempt to log in with different credentials. Positive test cases confirmed successful login with valid credentials, while negative test cases evaluated the system's response to invalid inputs. Invalid scenarios included incorrect username/password combinations, empty fields, and non-existent usernames. The test results demonstrated that the system appropriately prevented access upon encountering invalid credentials and provided clear error messages indicating the reason for login failure.

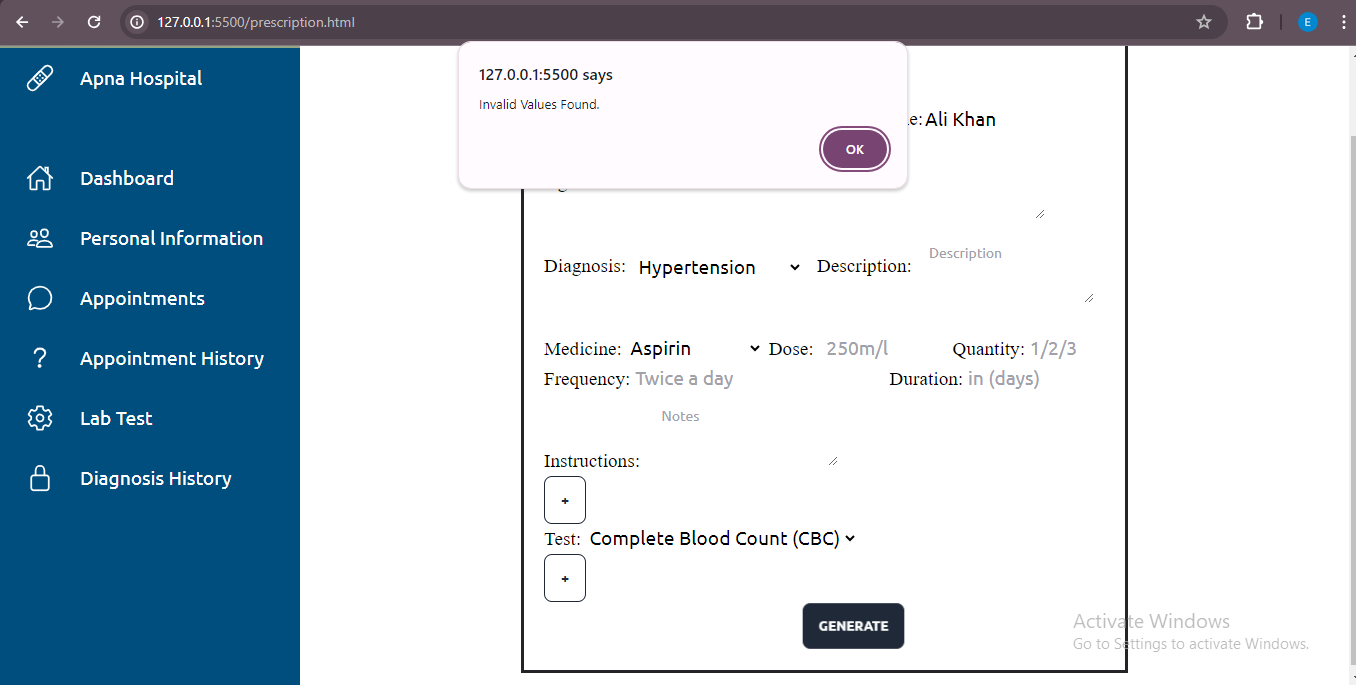
**Prescription Validation Testing:**

The prescription validation testing assessed the system's response to medication entries, ensuring that only valid inputs were accepted. Positive scenarios validated successful medication addition with accurate details. Negative cases assessed the system's reaction to incomplete or erroneous data, such as empty fields or non-numeric inputs in numeric fields. The testing confirmed that the system effectively enforced validation rules, preventing the addition of medications with invalid or incomplete information and prompting users with relevant error messages.

**Test Page Validation Testing:**

The test page validation testing focused on ensuring the integrity of numeric ID fields by rejecting alphabetic inputs. Positive tests verified the acceptance of valid numeric IDs, while negative scenarios attempted to input alphabetic characters into these fields. The results showed that the system correctly identified and rejected invalid inputs, maintaining the integrity of numeric ID fields.





## CONCLUSION

In conclusion, the Doctor Management Module represents a significant advancement in healthcare technology, aimed at enhancing the efficiency, effectiveness, and quality of patient care within healthcare facilities. By providing doctors with a comprehensive platform for managing appointments, accessing patient information, and facilitating medical interactions, the module streamlines workflows, improves communication, and empowers healthcare providers to deliver personalized and timely care to patients.

Through the integration of robust backend systems, user-friendly frontend interfaces, and secure authentication mechanisms, the module ensures seamless operation, data integrity, and confidentiality while adhering to modern design principles and industry standards. The utilization of a technology stack comprising HTML, CSS, JavaScript, Node.js, and MS SQL Server enables the development of a scalable, reliable, and secure platform capable of meeting the evolving needs of healthcare organizations and practitioners.

With features such as appointment management, prescription generation, lab test retrieval, and diagnosis history access, the module equips doctors with the tools and resources necessary for informed decision-making, efficient workflow management, and patient engagement. By centralizing patient data, facilitating collaboration among healthcare professionals, and promoting evidence-based practices, the module contributes to improved patient outcomes, enhanced patient satisfaction, and optimized resource utilization within healthcare settings.

In summary, the Doctor Management Module serves as a testament to the transformative potential of technology in healthcare, demonstrating how innovative solutions can empower healthcare providers, improve patient experiences, and advance the delivery of healthcare services in today's digital age. As healthcare continues to evolve, the module stands ready to adapt and evolve alongside it, driving continuous innovation and improvement in the pursuit of better health outcomes for all.