

# **CAPSTONE PROJECT**

## **HOSPITAL MANAGEMENT SYSTEM**



**Project by,**

**C.KARTHICK**

**JAVA FULLSTACK - (FACEPREP)**

**43120272**

**B.TECH – IT**

# **1. INTRODUCTION**

Hospitals handle a large amount of data daily, including patient records, doctor information, appointment schedules, and administrative tasks. Traditional hospital management often relies on manual record keeping, paper files, and disconnected systems. These methods can cause problems like data duplication, delays in accessing information, communication issues between departments, and a greater chance of human error. As healthcare services become more complex, there is a strong need for a centralized, automated system that can effectively manage hospital operations and improve service quality. The Hospital Management System (HMS) is a web-based application designed to automate and simplify the main functions of a hospital in an organized and user-friendly way. The system provides role-based access for Admin, Doctor, and Patient users, ensuring secure and authorized use. Patients can register, book appointments, and track their appointment status in real time. Doctors can manage their availability, see their assigned appointments, and approve or reject requests with proper explanations. Administrators manage the whole system by overseeing doctors, patients, and appointments while tracking hospital activities through analytical dashboards. Built using Java Full Stack technologies, the system uses the Model-View-Controller (MVC) architecture. This separates business logic, user interface, and data handling. This organized approach improves scalability, maintainability, and security. By digitizing hospital workflows and allowing real-time data access, the Hospital Management System greatly boosts operational efficiency, cuts down on manual work, and aids in effective decision-making, making it suitable for real hospital settings.

# **2. ABSTRACT**

The Hospital Management System is a Java Full Stack web application designed using Spring Boot, JSP, and MySQL to manage hospital operations efficiently. The system provides a secure and structured platform where users can log in based on their roles as Admin, Doctor, or Patient. New patients can register themselves, while existing users can authenticate using their credentials.

The application implements CRUD (Create, Read, Update, Delete) operations across all major modules such as patient management, doctor management, and appointment management. Patients can search for doctors, book appointments, and view appointment status. Administrators review appointment requests, manage doctors and patients, and approve or reject appointments. Doctors can manage their availability and further approve or reject appointments with a valid reason. The system also provides an analytical dashboard that visualizes appointment data using charts, aiding administrative decision-making.

### **3. OBJECTIVES OF THE PROJECT**

The primary objectives of the Hospital Management System are:

- To design a centralized platform for managing hospital activities
- To implement CRUD operations for patients, doctors, and appointments
- To provide role-based authentication and authorization
- To automate the appointment booking and approval process
- To allow doctors to manage availability and appointment decisions
- To enable patients to track appointment status transparently
- To generate analytical insights for hospital administration

### **4.SCOPE**

The scope of this project includes managing patient registration, doctor details, appointment booking, appointment approval workflows, and administrative monitoring. The system is suitable for small to medium-sized hospitals, clinics, and healthcare centers. Advanced modules such as billing, electronic medical records, and pharmacy management are not included but can be added as future enhancements.

### **5. TARGET USERS**

#### **Patients**

Patients can register, log in, view departments, search for doctors, book appointments, and track the status of their appointments.

#### **Doctors**

Doctors can log in to view appointments approved by the admin, update availability status, and approve or reject appointments with a reason.

#### **Administrators**

Admins manage the entire system, including doctors, patients, appointments, and analytics dashboards.

## 6.TECH STACK

| Category              | Technology Used         | Purpose / Description   |
|-----------------------|-------------------------|---|
| Programming Language  | Java                    | Core backend logic and business processing                    |
| Backend Framework     | Spring Boot             | Application framework for building scalable Java applications |
| Web Framework         | Spring MVC              | Handles HTTP requests and responses using MVC pattern         |
| ORM Framework         | Hibernate / JPA         | Manages database operations and object–relational mapping     |
| Frontend Technologies | JSP, HTML, CSS          | Used for designing and rendering user interface               |
| Database              | MySQL                   | Relational database for storing hospital data                 |
| Build Tool            | Maven                   | Dependency management and project build automation            |
| IDE                   | Spring Tool Suite (STS) | Development environment for Spring Boot applications          |
| Testing Tool          | Postman                 | Used for testing APIs and request handling                    |
| Server                | Embedded Tomcat         | Runs the Spring Boot web application                          |

Table 1: Tech Stack

## 7. SYSTEM REQUIREMENTS

### HARDWARE REQUIREMENTS

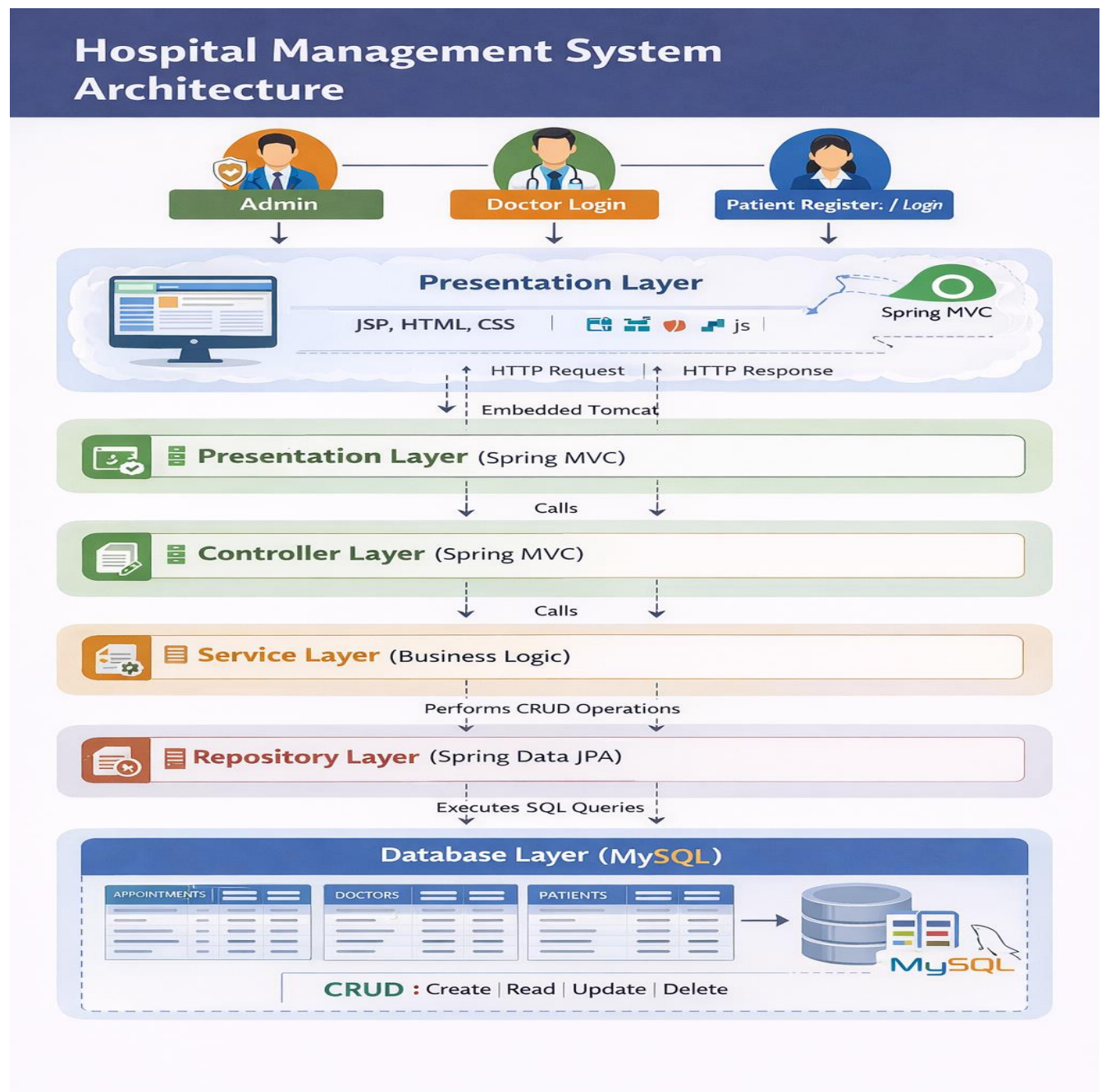
- **Processor:** Dual Core or higher
- **RAM:** Minimum 4 GB (Recommended 8 GB)
- **Hard Disk:** Minimum 5 GB free space
- **Display Resolution:** 1366 × 768 or higher
- **Internet Connection:** Required for development and deployment

### SOFTWARE REQUIREMENTS

- **Operating System:** Windows 10 / Windows 11
- **Programming Language:** Java JDK 8 or above
- **Backend Framework:** Spring Boot 3.x
- **Web Framework:** Spring MVC (included with Spring Boot)
- **ORM Framework:** Hibernate / JPA (latest stable version)
- **Database:** MySQL Server 8.0 or above
- **IDE:** Spring Tool Suite (STS) 4.x
- **Build Tool:** Maven 3.6 or above
- **Application Server:** Embedded Apache Tomcat (comes with Spring Boot)
- **Frontend Technologies:** JSP, HTML5, CSS3
- **Web Browser:** Google Chrome (latest) / Microsoft Edge (latest)
- **API Testing Tool:** Postman (latest version)

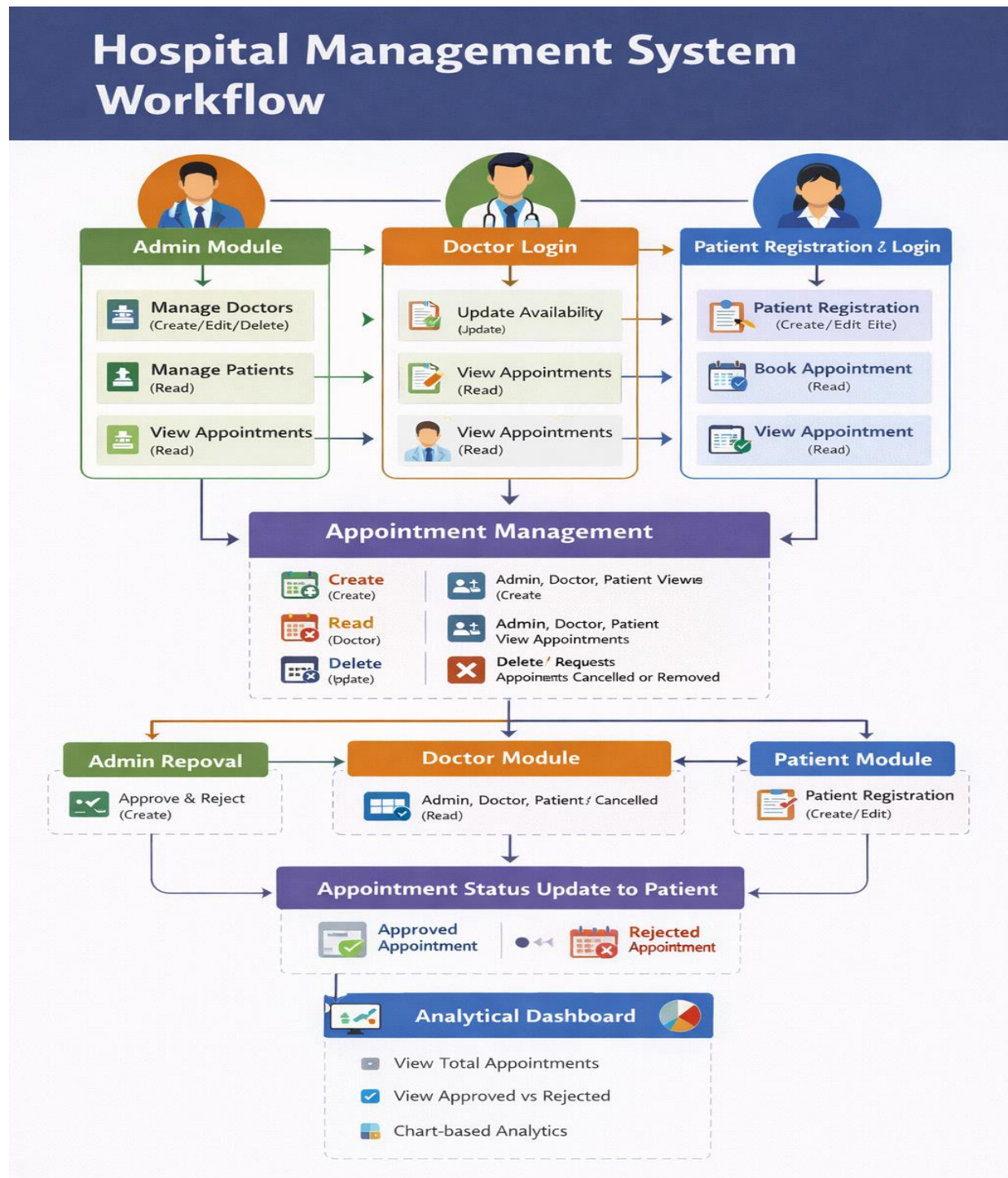
## 8.SYSTEM ARCHITECTURE

The Presentation Layer provides the user interface for Admin, Doctor, and Patient using JSP, HTML, CSS, and JavaScript, and handles user requests. The Controller Layer processes HTTP requests and routes them to appropriate services using Spring MVC. The Service Layer contains the core business logic and manages application workflows, including appointment handling and validations. The Repository and Database Layers use Spring Data JPA to perform CRUD operations and store all hospital data securely in a MySQL database.



## 9.SYSTEM WORKFLOW

This diagram shows the workflow of the Hospital Management System with Admin, Doctor, and Patient roles. It illustrates appointment booking, approval or rejection, and real-time status updates. The system supports CRUD operations and centralized appointment management. An analytical dashboard displays appointment statistics and insights.



## 10.DATABASE DESIGN

The database is designed using normalization principles to reduce redundancy and maintain data integrity. MySQL is used as the relational database management system.

### Database Tables:

- **Appointments:** Stores the appointments booked by the patients
- **Doctors:** Stores the list of doctors working in the hospital
- **Patients:** Stores the list of patients enrolled for treatment in the hospital
- **Users:** Stores all the users including Admin, Doctors, Patients

Each table is related through primary and foreign keys to maintain relational integrity.

The screenshot displays the MySQL Workbench interface for a local instance of MySQL 8.0. The main editor window shows the following SQL script:

```
1 CREATE DATABASE hospital_db;
2 USE hospital_db;
3 show tables;
4 CREATE TABLE users (
5     user_id BIGINT PRIMARY KEY AUTO_INCREMENT,
6     username VARCHAR(50) UNIQUE NOT NULL,
7     password VARCHAR(100) NOT NULL,
8     role VARCHAR(20) NOT NULL, -- ADMIN / DOCTOR / PATIENT
9     reference_id BIGINT -- doctor_id or patient_id
10 );
11 INSERT INTO users (username, password, role)
```

The left sidebar shows the 'MANAGEMENT' tab with a tree view of the database structure. The 'Tables\_in\_hospital\_db' table is selected, showing a list of tables: appointments, doctors, patients, and users. The 'Output' tab at the bottom shows the execution results of the SQL script, including the creation of the database, the creation of the users table, and the insertion of data into the users table.

| #  | Time     | Action                                   | Message  | Duration / Fetch      |
|----|----------|--|--|-----------------------|
| 7  | 10:44:07 | select * from doctors LIMIT 0, 1000      | 3 row(s) returned                                      | 0.000 sec / 0.000 sec |
| 8  | 10:44:11 | select * from patients LIMIT 0, 1000     | 6 row(s) returned                                      | 0.000 sec / 0.000 sec |
| 9  | 10:46:33 | ALTER TABLE patients DROP COLUMN disease | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 | 0.312 sec             |
| 10 | 10:46:36 | select * from patients LIMIT 0, 1000     | 6 row(s) returned                                      | 0.000 sec / 0.000 sec |
| 11 | 21:54:59 | show tables                              | 4 row(s) returned                                      | 0.000 sec / 0.000 sec |

## 11. HTTP REQUEST METHODS & API DOCUMENTATION

The Hospital Management System follows RESTful architecture principles and uses standard HTTP methods to perform CRUD operations on hospital data such as users, patients, doctors, and appointments. Each API endpoint is designed to handle a specific operation and returns responses in JSON format.

### 11.1 API Mapping Table

| Method | Endpoint                          | Description  |
|--------|-----------------------------------|--|
| POST   | /auth/login                       | Authenticates Admin or Doctor and returns role details     |
| POST   | /auth/register                    | Registers a new patient in the system                      |
| GET    | /patients                         | Retrieves the list of all registered patients (Admin only) |
| GET    | /patients/{id}                    | Fetches details of a specific patient                      |
| POST   | /patients                         | Creates a new patient record                               |
| GET    | /doctors                          | Retrieves the list of all doctors                          |
| POST   | /doctors                          | Adds a new doctor (Admin only)                             |
| PUT    | /doctors/{id}/availability        | Updates doctor availability status                         |
| GET    | /appointments                     | Retrieves all appointment records (Admin view)             |
| GET    | /appointments/patient/{id}        | Retrieves appointments for a specific patient              |
| GET    | /appointments/doctor/{id}         | Retrieves appointments assigned to a doctor                |
| POST   | /appointments                     | Books a new appointment (Patient)                          |
| PUT    | /appointments/{id}/admin-approve  | Approves or rejects appointment by Admin                   |
| PUT    | /appointments/{id}/doctor-approve | Accepts or rejects appointment by Doctor with reason       |
| GET    | /analytics/appointments           | Fetches appointment statistics for admin dashboard         |

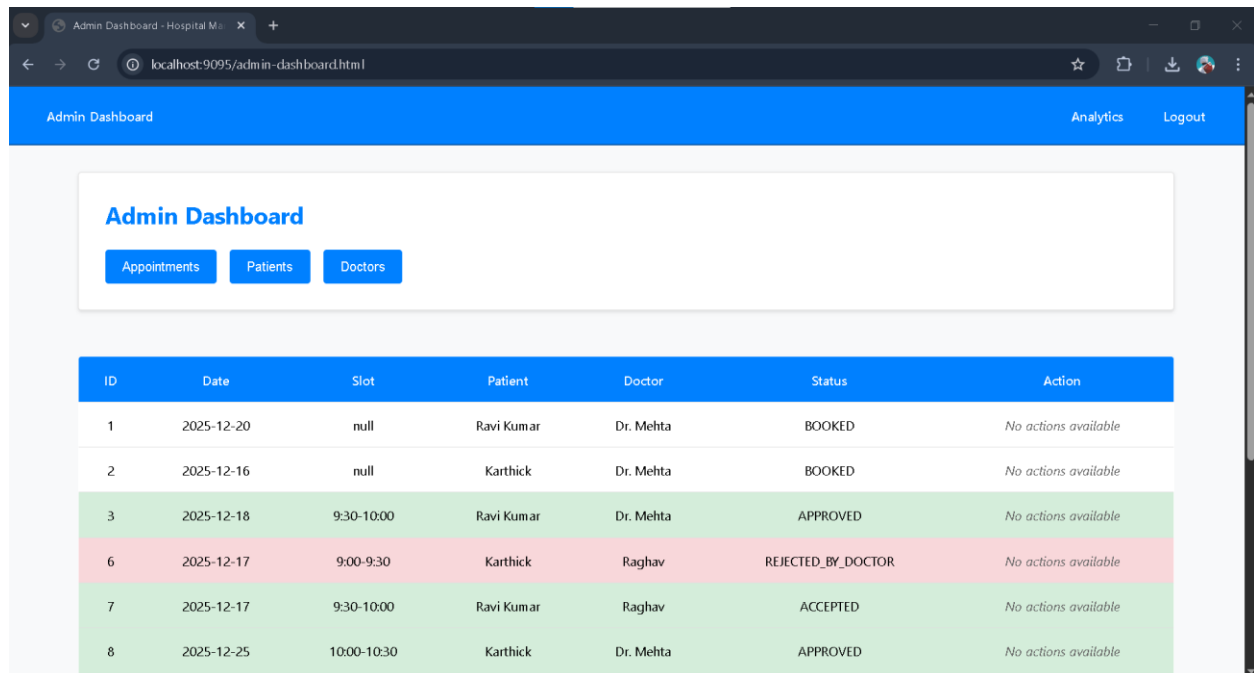
### 11.2 HTTP Methods Usage Summary

- **POST** – Used to create new records such as patients, doctors, and appointments
- **GET** – Used to retrieve data from the system
- **PUT** – Used to update appointment status and doctor availability
- **DELETE** – Used to remove records (optional / future scope)

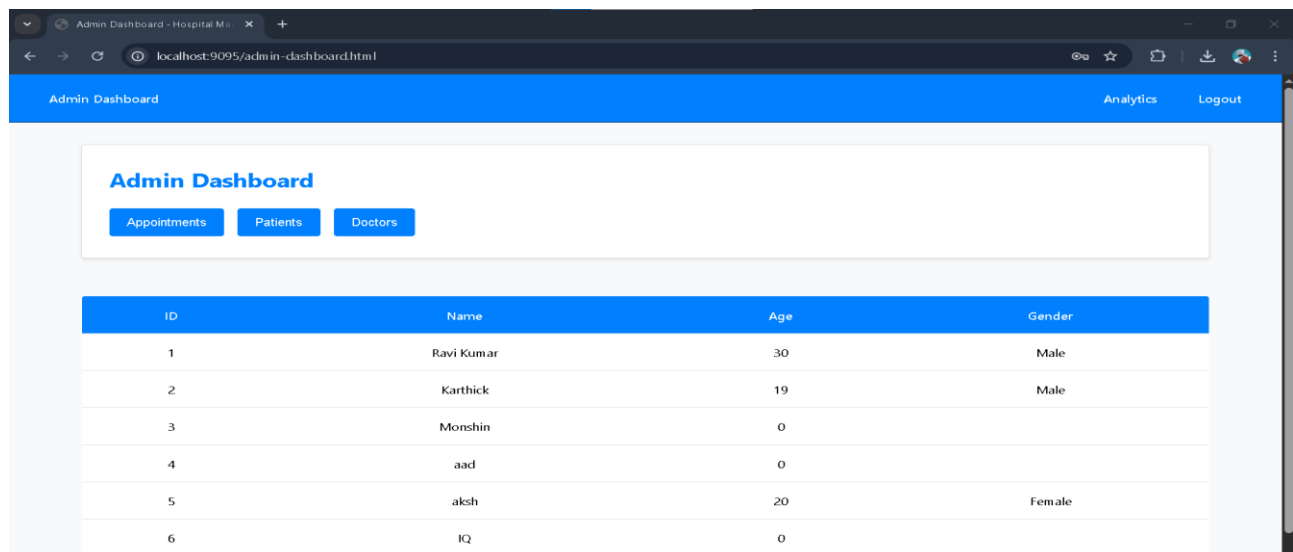
## 12 PROJECT MODULES

### 12.1 Admin Module:

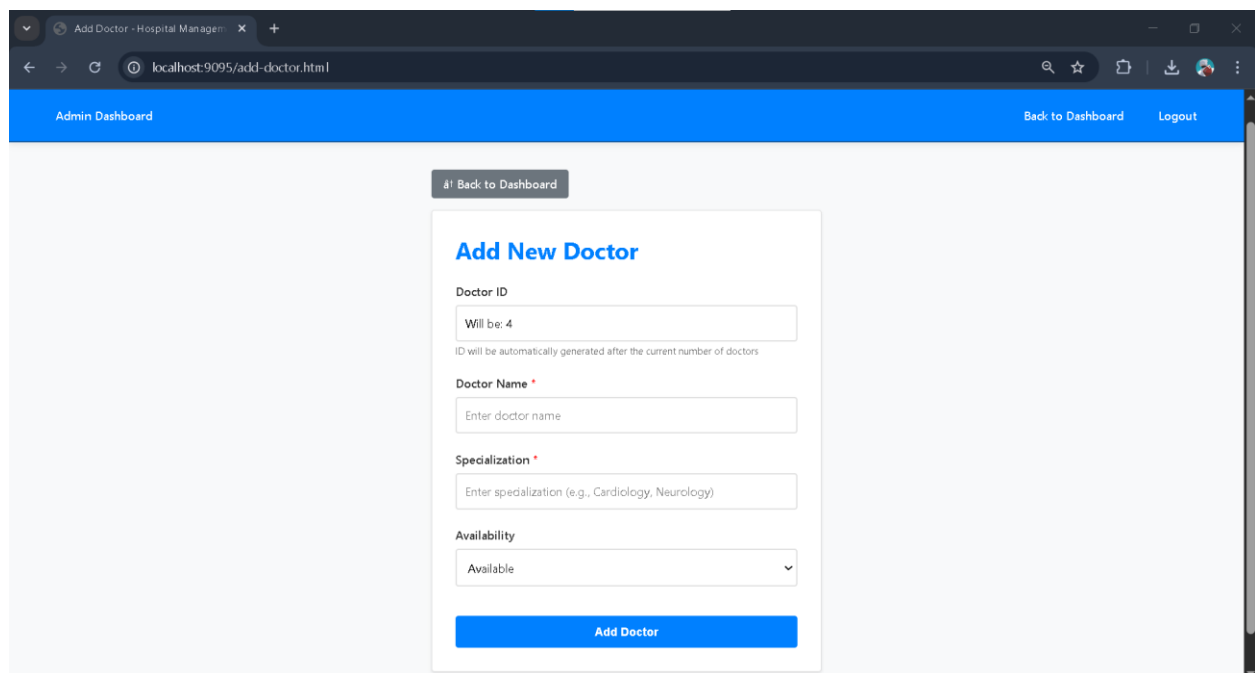
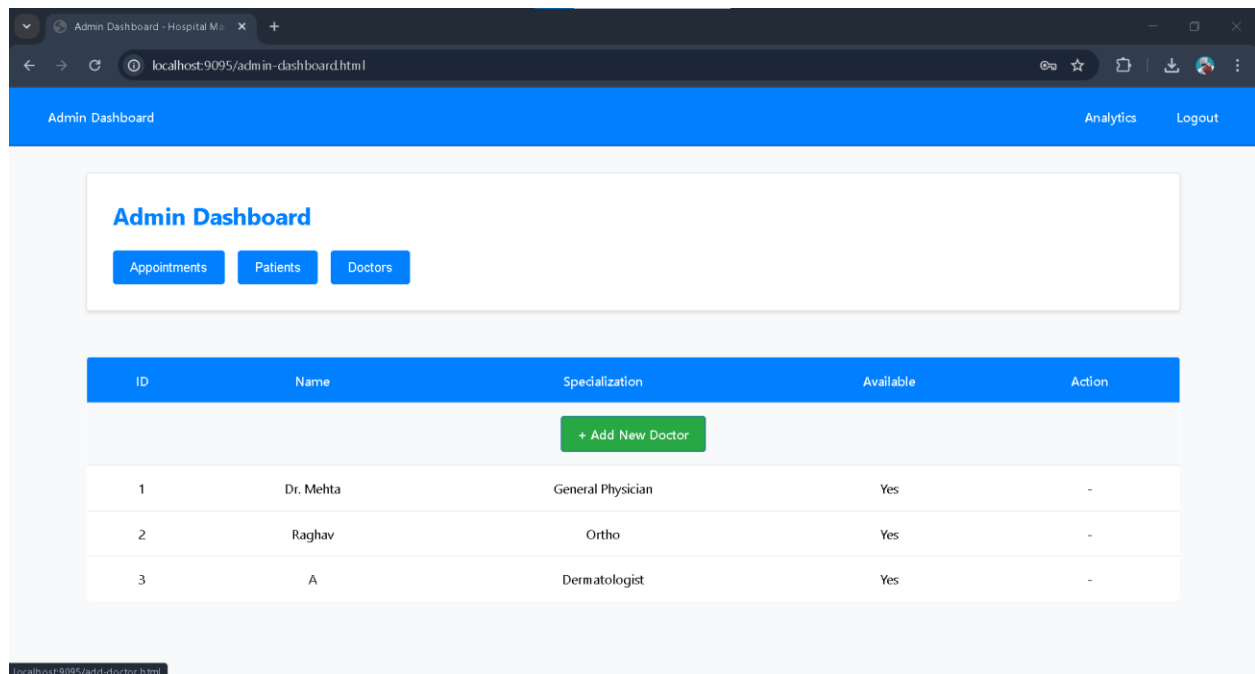
The Admin Module serves as the main control center of the Hospital Management System. It lets the administrator view and manage all appointment requests, as well as details about patients and doctors. The admin can track appointment statuses, including booked, approved, accepted, or rejected, in real time. They can also add a new doctor. The module offers easy access to patient and doctor management sections and includes a dashboard for visualizing appointment data. This module helps ensure smooth coordination and efficient oversight of hospital operations.



| ID | Date       | Slot        | Patient    | Doctor    | Status             | Action               |
|----|------------|-------------|------------|-----------|--------------------|----------------------|
| 1  | 2025-12-20 | null        | Ravi Kumar | Dr. Mehta | BOOKED             | No actions available |
| 2  | 2025-12-16 | null        | Karthick   | Dr. Mehta | BOOKED             | No actions available |
| 3  | 2025-12-18 | 9:30-10:00  | Ravi Kumar | Dr. Mehta | APPROVED           | No actions available |
| 6  | 2025-12-17 | 9:00-9:30   | Karthick   | Raghav    | REJECTED_BY_DOCTOR | No actions available |
| 7  | 2025-12-17 | 9:30-10:00  | Ravi Kumar | Raghav    | ACCEPTED           | No actions available |
| 8  | 2025-12-25 | 10:00-10:30 | Karthick   | Dr. Mehta | APPROVED           | No actions available |

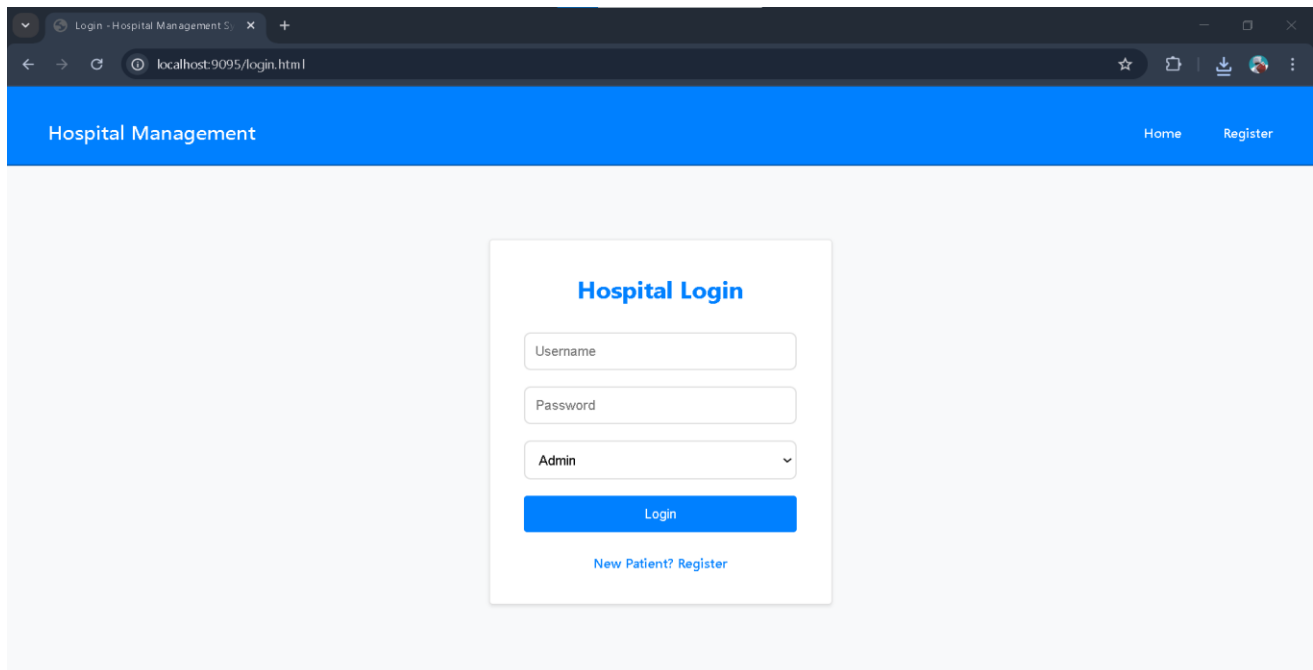


| ID | Name       | Age | Gender |
|----|------------|-----|--------|
| 1  | Ravi Kumar | 30  | Male   |
| 2  | Karthick   | 19  | Male   |
| 3  | Monshin    | 0   |        |
| 4  | aad        | 0   |        |
| 5  | aksh       | 20  | Female |
| 6  | IQ         | 0   |        |



## 12.2 Login Module:

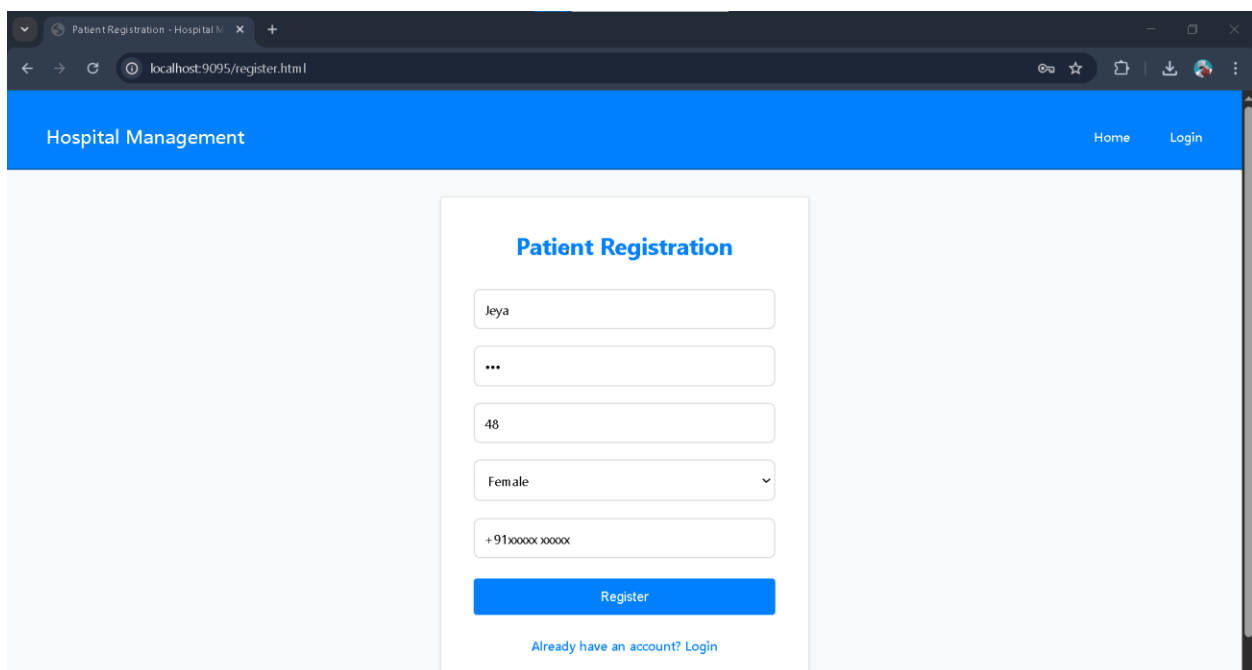
The Login Module lets Admin and Doctor users log in with their predefined credentials to reach their dashboards. Patients can register as new users or log in using their existing credentials through the same interface. The system checks user details based on their selected role and redirects them as needed. This module provides role-based access and secure entry into the Hospital Management System.



The screenshot shows a web browser window with the address bar displaying 'localhost:9095/login.html'. The page has a blue header with 'Hospital Management' on the left and 'Home' and 'Register' links on the right. The main content area features a white box titled 'Hospital Login'. Inside this box, there are three input fields: 'Username', 'Password', and a dropdown menu currently showing 'Admin'. Below these fields is a blue 'Login' button. At the bottom of the box, there is a link that says 'New Patient? Register'.

### 12.3 Register Module:

The Patient Registration Module lets new patients set up an account in the Hospital Management System by providing personal and login details. The system securely keeps patient information like name, age, gender, and contact number in the database. After registration, patients can log in with their credentials to access the patient dashboard. This module helps manage patient data effectively and allows for easy appointment booking and tracking.



The screenshot shows a web browser window with the address bar displaying 'localhost:9095/register.html'. The page has a blue header with 'Hospital Management' on the left and 'Home' and 'Login' links on the right. The main content area features a white box titled 'Patient Registration'. Inside this box, there are five input fields: a text field with 'Jeya', a text field with '...', a text field with '48', a dropdown menu showing 'Female', and a text field with '+91xxxxxx xxxxx'. Below these fields is a blue 'Register' button. At the bottom of the box, there is a link that says 'Already have an account? Login'.

## 12.4 Doctor Module:

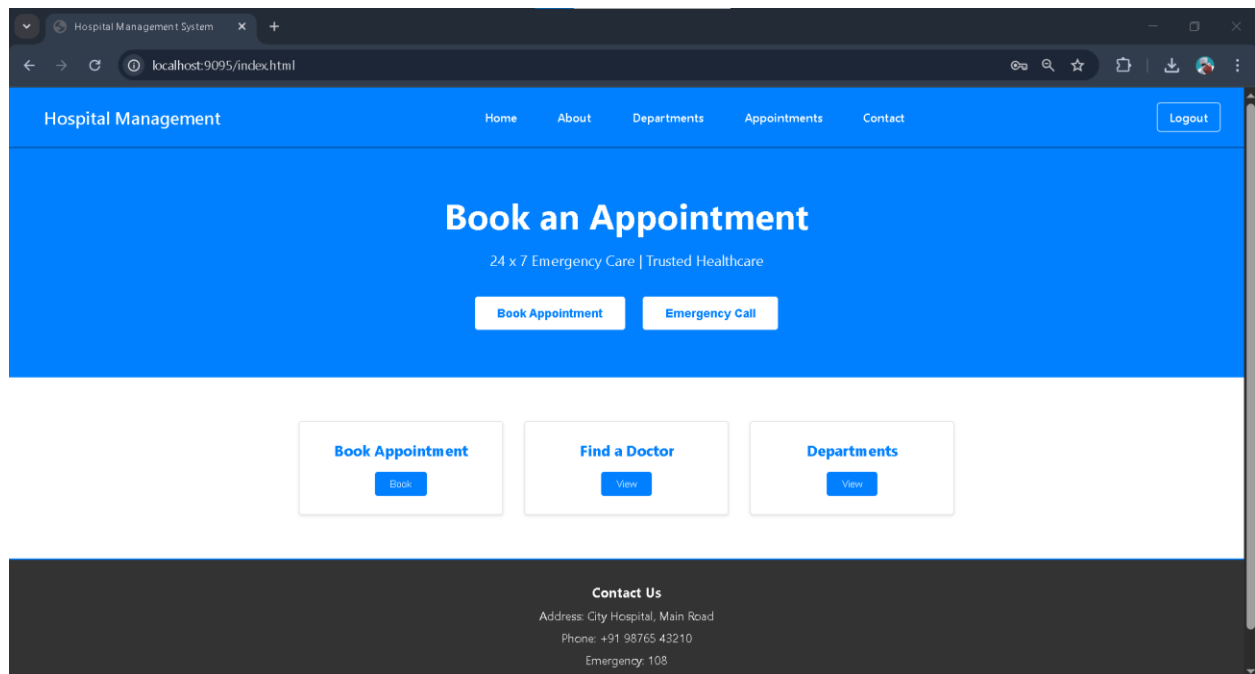
The Doctor Module lets doctors see all appointment requests assigned to them on a clear dashboard. Doctors can change their availability status. This status determines if patients can book appointments. Doctors can accept or reject appointments based on requests approved by the admin. If an appointment is rejected, the doctor must include a reason, which the patient can see. The module also offers real-time updates on appointment status and ensures efficient scheduling between doctors and patients.

The screenshot displays the 'Doctor Dashboard' interface in a web browser. The browser's address bar shows 'localhost:9095/doctor-dashboard.html'. The dashboard has a blue header with the title 'Doctor Dashboard' and a 'Logout' button. Below the header, there is a section with a 'Refresh Appointments' button and an 'Availability' toggle switch set to 'ON'. The main part of the dashboard is a table with the following data:

| ID | Date       | Slot        | Patient    | Status             | Action               |
|----|------------|-------------|------------|--------------------|----------------------|
| 6  | 2025-12-17 | 9:00-9:30   | Karthick   | REJECTED_BY_DOCTOR | No actions available |
| 7  | 2025-12-17 | 9:30-10:00  | Ravi Kumar | ACCEPTED           | No actions available |
| 12 | 2025-12-24 | 10:30-11:00 | aksh       | ACCEPTED           | No actions available |
| 13 | 2025-10-24 | 10:00-10:30 | aksh       | REJECTED_BY_DOCTOR | No actions available |
| 14 | 2026-01-05 | 9:30-10:00  | aksh       | ACCEPTED           | No actions available |

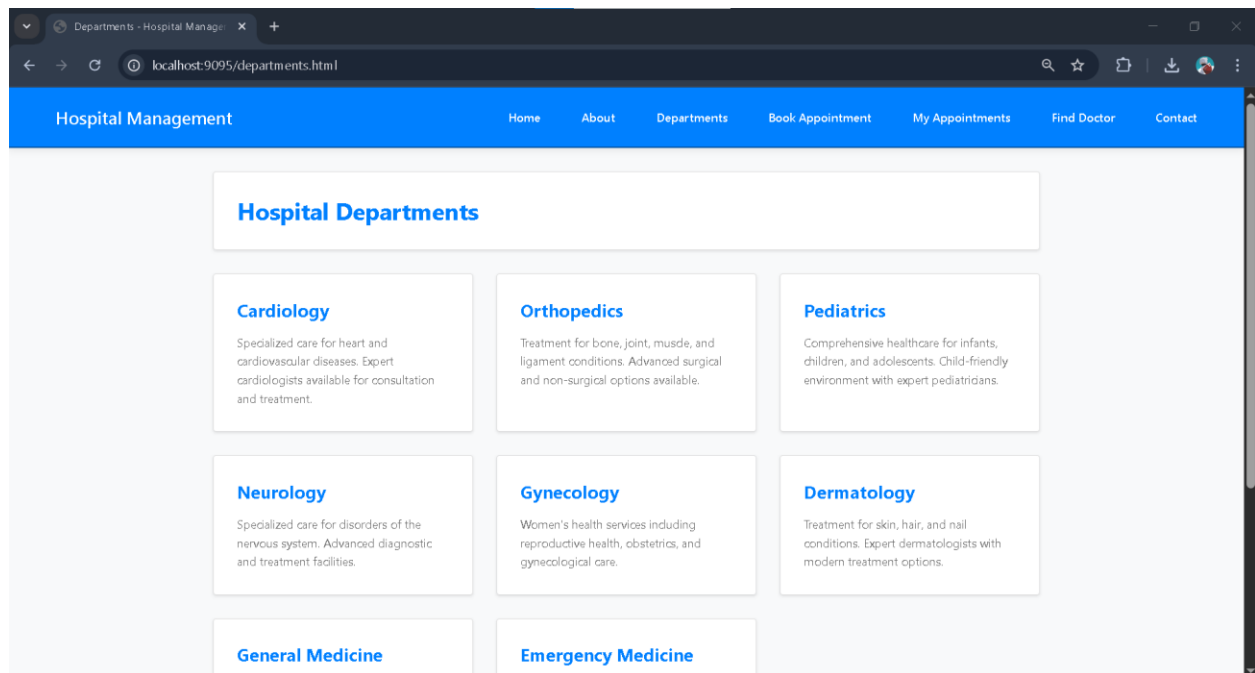
## 12.5 Patient Home & Appointment Booking Module

The Patient Home Page is the main interface for patients after they log in. It lets patients book appointments, find available doctors, and view hospital departments through easy navigation options. The module also gives quick access to emergency contact services. With a user-friendly dashboard, this module promotes smooth patient interaction and efficient appointment initiation within the Hospital Management System.



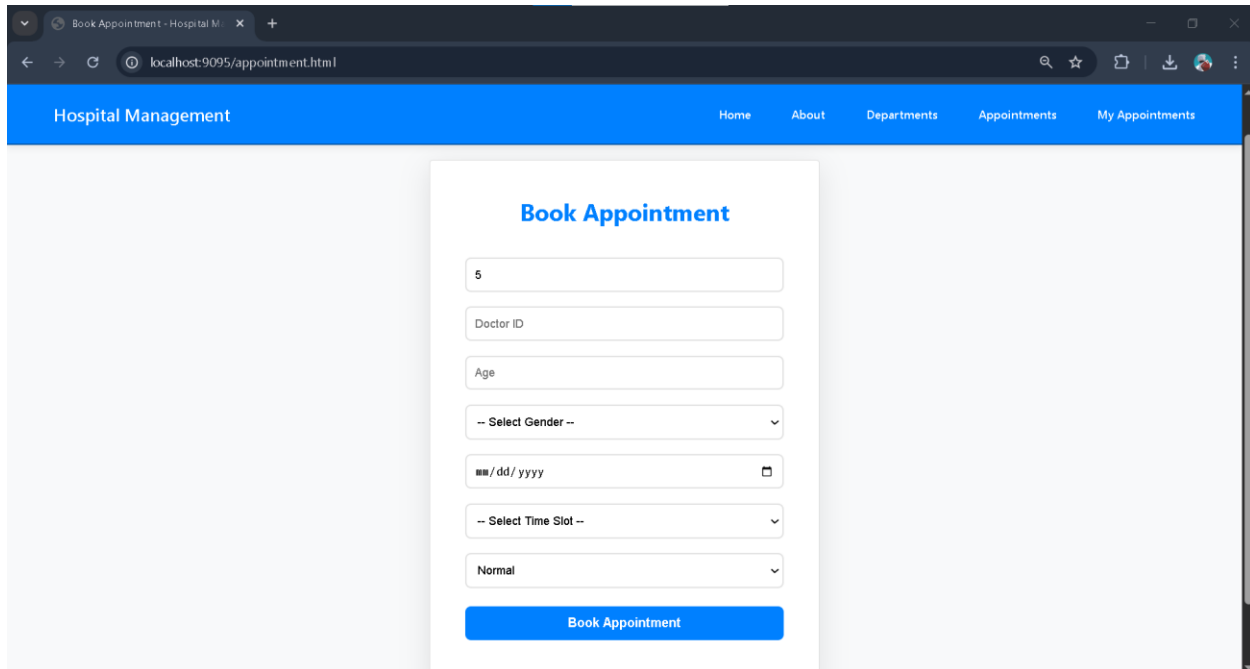
## 12.6 Departments Module:

The Departments Module gives a clear overview of the different medical departments in the hospital. It helps patients see the treatments and services each department offers, including Cardiology, Orthopedics, Pediatrics, and others. This module aids patients in selecting the right department before they book an appointment. By organizing medical services, it makes navigation easier and improves the overall patient experience.



## 12.7 Appointment Booking Module:

The Appointment Booking Module allows patients to schedule appointments by choosing a doctor, preferred date, time slot, and appointment type. It collects patient details like age and gender for proper record keeping. After booking, the request goes for admin and doctor approval. The appointment status updates in real-time, and patients can track it in the “My Appointments” section.



The screenshot shows a web browser window with the URL `localhost:9095/appointment.html`. The page has a blue header with the text "Hospital Management" and navigation links: "Home", "About", "Departments", "Appointments", and "My Appointments". The main content area features a "Book Appointment" form. The form includes the following fields and controls:

- A text input field containing the number "5".
- A text input field labeled "Doctor ID".
- A text input field labeled "Age".
- A dropdown menu labeled "-- Select Gender --".
- A date input field with a calendar icon and the placeholder text "dd / dd / yyyy".
- A dropdown menu labeled "-- Select Time Slot --".
- A dropdown menu with the option "Normal" selected.
- A blue button labeled "Book Appointment".

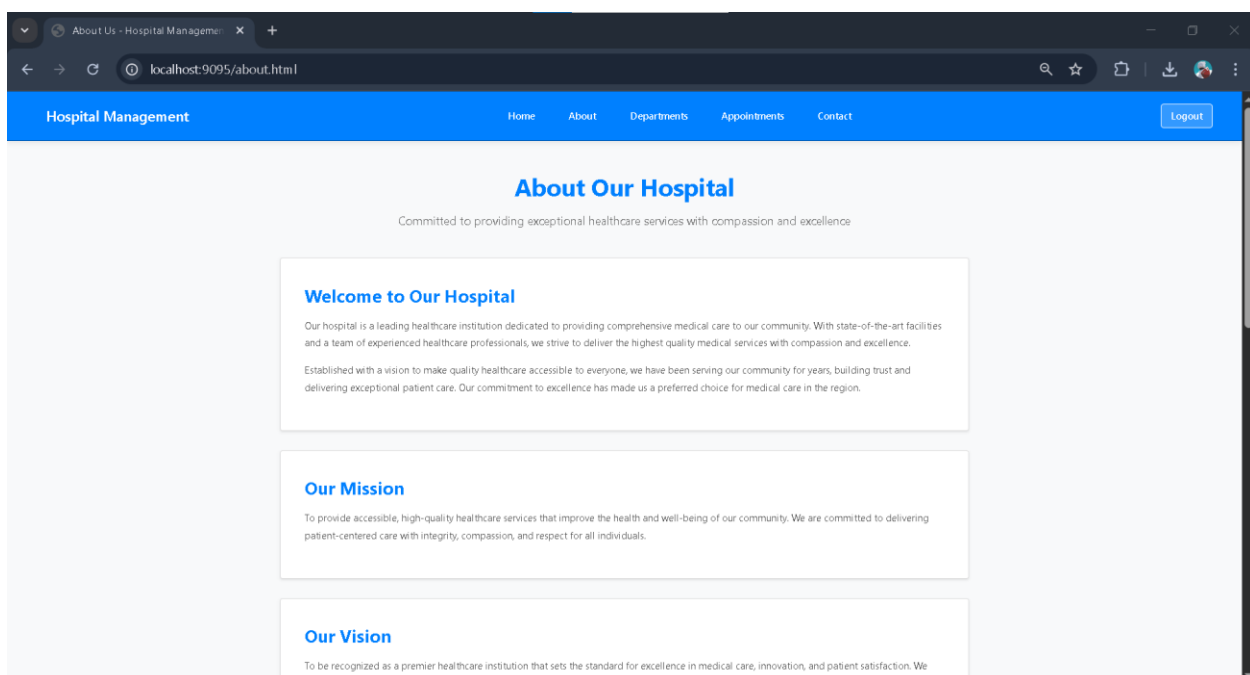
## 12.8 My Appointments Module:

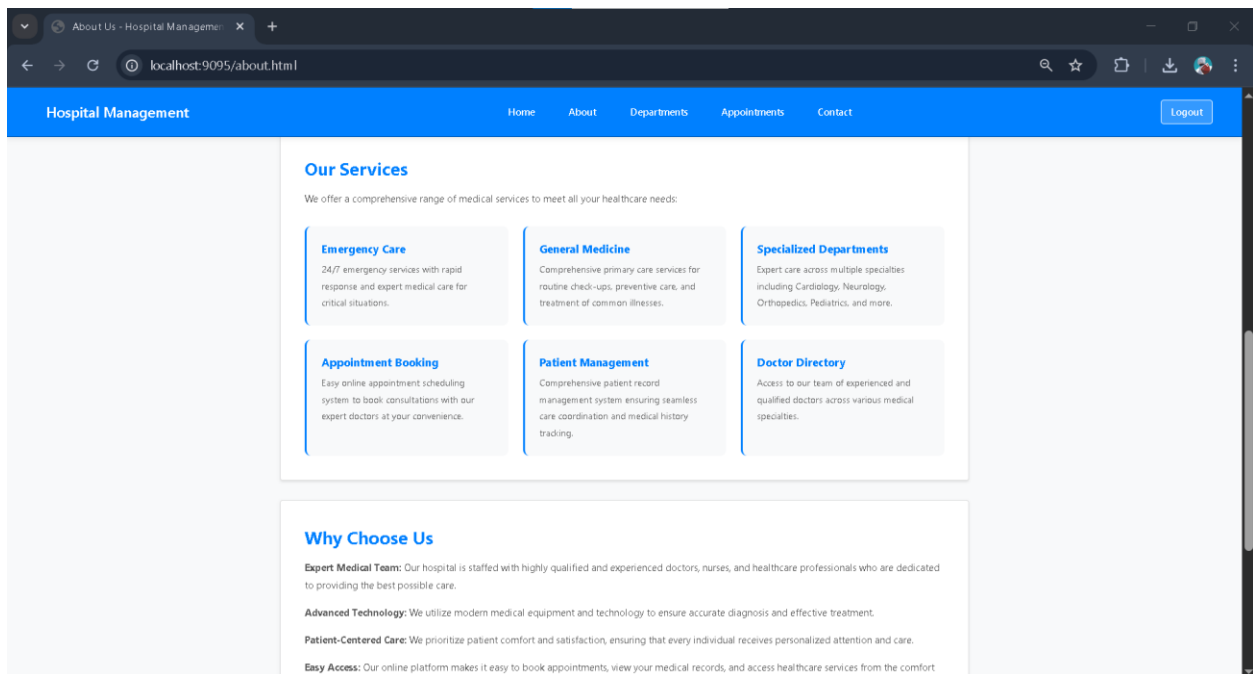
The My Appointments Module lets patients see all their booked appointments clearly and in an organized way. It shows key details like appointment date, time, doctor name, priority, current status, and notes. Patients can check if an appointment is approved by the admin or accepted or rejected by the doctor, along with any reasons for rejection. This module offers clarity and keeps patients updated about their appointment status in real time.

| ID | Date       | Slot        | Doctor    | Priority  | Status                                 | Remarks                         |
|----|------------|-------------|-----------|-----------|--|---------------------------------|
| 10 | 2025-12-25 | 10:30-11:00 | Dr. Mehta | NORMAL    | Approved by Admin (Waiting for Doctor) | -                               |
| 11 | 2025-12-26 | 9:30-10:00  | A         | EMERGENCY | Accepted by Doctor                     | Appointment confirmed           |
| 12 | 2025-12-24 | 10:30-11:00 | Raghav    | EMERGENCY | Accepted by Doctor                     | Appointment confirmed           |
| 13 | 2025-10-24 | 10:00-10:30 | Raghav    | NORMAL    | Rejected by Doctor                     | I've have a surgery on that day |
| 14 | 2026-01-05 | 9:30-10:00  | Raghav    | EMERGENCY | Accepted by Doctor                     | Appointment confirmed           |
| 15 | 2025-12-25 | 10:00-10:30 | A         | NORMAL    | Rejected by Doctor                     | have a surgery                  |

## 12.9 About Module:

The About Module offers an overview of the hospital, including its mission, vision, and healthcare values. It emphasizes the hospital's dedication to quality medical services, patient-centered care, and modern healthcare facilities. This module also lists the services available and explains why patients should select this hospital. It helps build trust and provides users with a clear understanding of the hospital's goals and abilities.





## 13.DATABASE TABLES

### 13.1 Users Table:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

Server: Local instance MySQL80

Client: Local instance MySQL80

Users: Local instance MySQL80

Status: Local instance MySQL80

Data: Local instance MySQL80

INSTANCE

Startup: Local instance MySQL80

Server: Local instance MySQL80

Option: Local instance MySQL80

Admin: Local instance MySQL80

Information: Local instance MySQL80

SQL Editor

```

8  role VARCHAR(20) NOT NULL, -- ADMIN / DOCTOR / PATIENT
9  reference_id BIGINT -- doctor_id or patient_id
10 );
11 * INSERT INTO users (username, password, role)
12 VALUES ('admin', '123', 'ADMIN'), ('raghav', '123', 'DOCTOR'), ('mehta', '123', 'DOCTOR');
13 * INSERT INTO users (username, password, role)
14 VALUES ('anup', '123', 'doctor');
15 * INSERT INTO users (username, password, role)
16 VALUES ('A', '123', 'DOCTOR');
17 * select * from users;
18 * drop table users;
19 * SELECT appointment_id, status, rejection_reason FROM appointments;

```

Result Grid

|   | user_id | username | password | role | reference_id |
|---|---------|----------|----------|------|--------------|
| 1 | admin   | 123      | ADMIN    |      |              |
| 2 | raghav  | 123      | DOCTOR   | 2    |              |
| 3 | mehta   | 123      | DOCTOR   | 3    |              |
| 4 | Monshin | 123      | PATIENT  | 3    |              |
| 5 | aad     | 1        | PATIENT  | 4    |              |
| 6 | aksh    | ak       | PATIENT  | 5    |              |
| 7 | anup    | 123      | doctor   |      |              |
| 8 | A       | 123      | DOCTOR   | 3    |              |
| 9 | IQ      | iq       | PATIENT  | 6    |              |

Output

Action Output

| #  | Time     | Action                                   | Message  | Duration / Fetch      |
|----|----------|--|--|-----------------------|
| 8  | 10:44:11 | select * from patients LIMIT 0,1000      | 6 row(s) returned                                      | 0.000 sec / 0.000 sec |
| 9  | 10:45:33 | ALTER TABLE patients DROP COLUMN disease | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 | 0.312 sec             |
| 10 | 10:46:36 | select * from patients LIMIT 0,1000      | 6 row(s) returned                                      | 0.000 sec / 0.000 sec |

SQL Editor

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

## 13.2 Appointments Table:

The screenshot shows the MySQL Workbench interface with the 'Appointments' table selected in the Navigator. The table structure is defined as follows:

| appointment_id | appointment_date | priority  | status             | doctor_id | patient_id | rejection_reason    | slot        |
|----------------|------------------|-----------|--------------------|-----------|------------|---------------------|-------------|
| 1              | 2025-12-20       | EMERGENCY | BOOKED             | 1         | 1          |                     |             |
| 2              | 2025-12-16       | NORMAL    | BOOKED             | 1         | 2          |                     |             |
| 3              | 2025-12-18       | EMERGENCY | APPROVED           | 1         | 1          |                     | 9:30-10:00  |
| 6              | 2025-12-17       | NORMAL    | REJECTED_BY_DOCTOR | 2         | 2          | surgery time        | 9:00-9:30   |
| 7              | 2025-12-17       | EMERGENCY | ACCEPTED           | 2         | 1          |                     | 9:30-10:00  |
| 8              | 2025-12-25       | NORMAL    | APPROVED           | 1         | 2          |                     | 10:00-10:30 |
| 9              | 2025-12-24       | EMERGENCY | APPROVED           | 1         | 3          |                     | 9:30-10:00  |
| 10             | 2025-12-25       | NORMAL    | APPROVED           | 1         | 5          |                     | 10:30-11:00 |
| 11             | 2025-12-26       | EMERGENCY | ACCEPTED           | 3         | 5          |                     | 9:30-10:00  |
| 12             | 2025-12-24       | EMERGENCY | ACCEPTED           | 2         | 5          |                     | 10:30-11:00 |
| 13             | 2025-10-24       | NORMAL    | REJECTED_BY_DOCTOR | 2         | 5          | I've have a surg... | 10:00-10:30 |
| 14             | 2026-01-05       | EMERGENCY | ACCEPTED           | 2         | 5          |                     | 9:30-10:00  |
| 15             | 2025-12-25       | NORMAL    | REJECTED_BY_DOCTOR | 3         | 5          | have a surgery      | 10:00-10:30 |

The Output pane shows the execution of two queries:

- Query 12: 22:45:13 select \* from users LIMIT 0, 1000. Message: 9 row(s) returned. Duration / Fetch: 0.000 sec / 0.000 sec.
- Query 13: 22:46:05 select \* from appointments LIMIT 0, 1000. Message: 13 row(s) returned. Duration / Fetch: 0.000 sec / 0.000 sec.

## 13.3 Doctors Table:

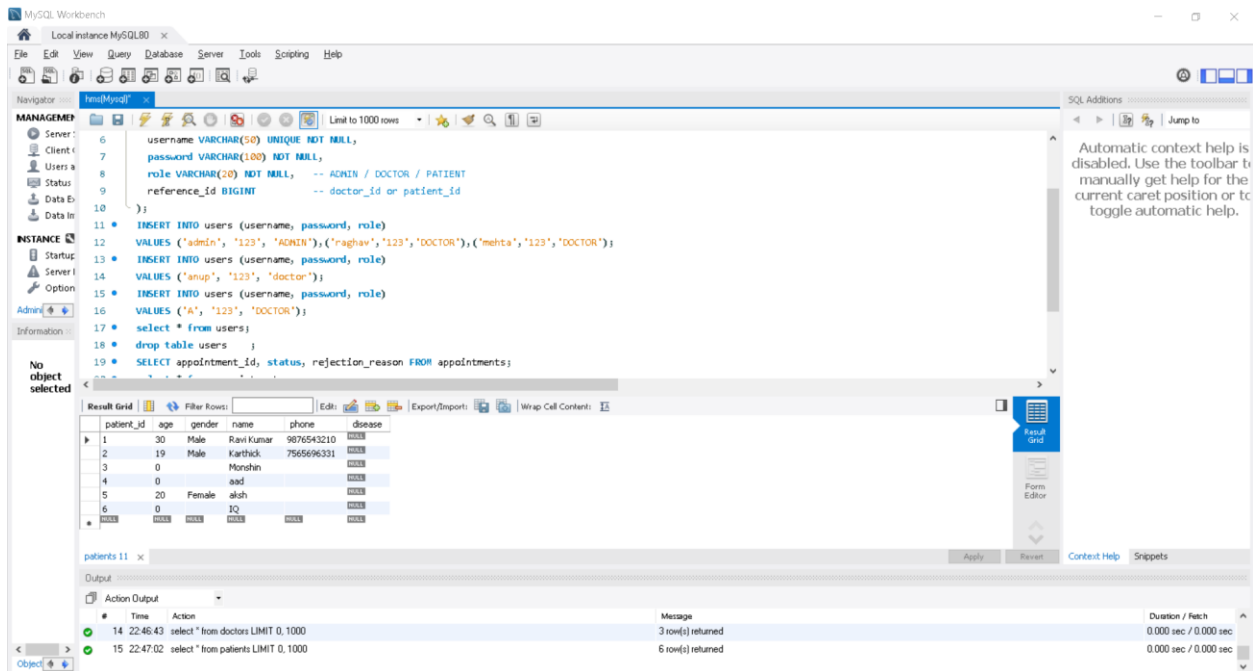
The screenshot shows the MySQL Workbench interface with the 'Doctors' table selected in the Navigator. The table structure is defined as follows:

| doctor_id | available | name      | specialization    |
|-----------|-----------|-----------|-------------------|
| 1         | 1         | Dr. Mehta | General Physician |
| 2         | 1         | Raghav    | Ortho             |
| 3         | 1         | A         | Dermatologist     |

The Output pane shows the execution of two queries:

- Query 13: 22:46:05 select \* from appointments LIMIT 0, 1000. Message: 13 row(s) returned. Duration / Fetch: 0.000 sec / 0.000 sec.
- Query 14: 22:46:43 select \* from doctors LIMIT 0, 1000. Message: 3 row(s) returned. Duration / Fetch: 0.000 sec / 0.000 sec.

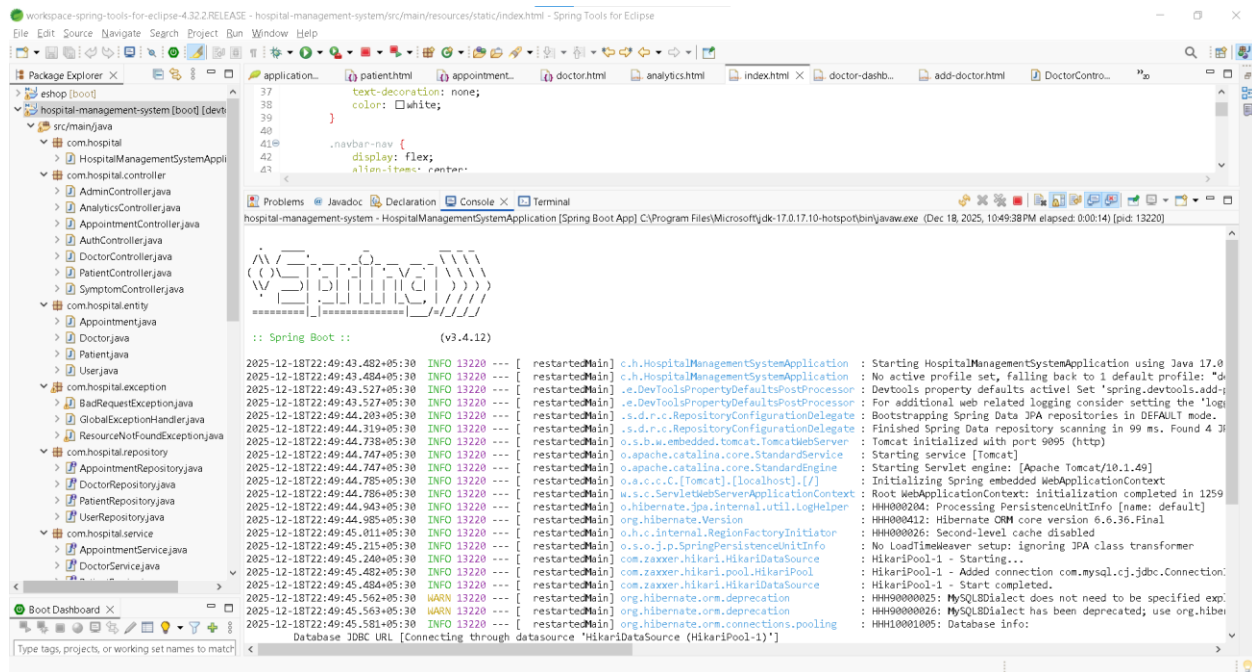
## 13.4 Patients Table:



## 14.TESTING

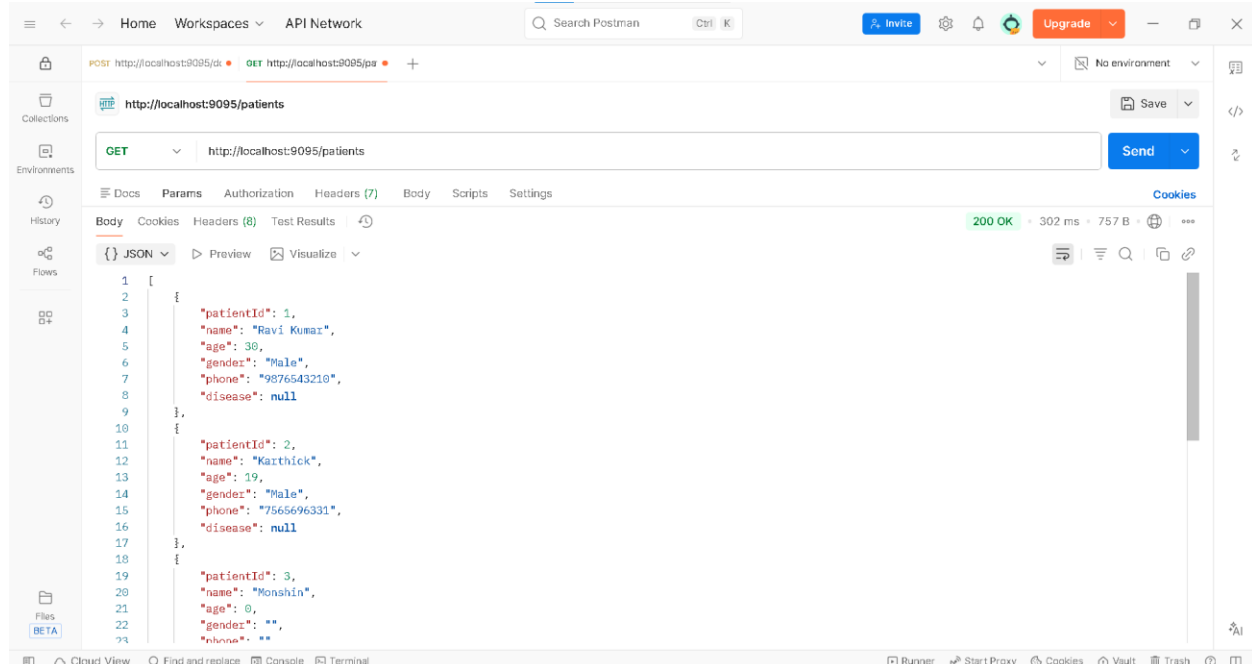
### 14.1 Spring Boot Testing:

This screenshot shows the successful running of the Hospital Management System with Spring Boot in Spring Tool Suite (STS). The console logs indicate that the application started without errors, initialized the embedded Tomcat server, and connected to the MySQL database using JPA and Hibernate. Spring Data JPA repositories are scanned and loaded correctly, which confirms the backend is set up properly. This shows that the application is running and ready to handle user requests through the web interface.



## 14.2 Postman API Testing:

Postman is used to test the REST APIs developed in the Hospital Management System. Using Postman, HTTP requests such as GET, POST, PUT, and DELETE are sent to the Spring Boot application to verify backend functionality. The responses are received in JSON format, confirming successful communication between the controller, service, and database layers. This testing ensures that CRUD operations are working correctly and validates the reliability of the RESTful web services.



## **15.CONCLUSION AND FUTURE ENHANCEMENTS**

### **15.1 CONCLUSION:**

In this project, I created a Hospital Management System as a Java Full Stack web application to automate and simplify hospital operations. The system replaces manual processes with a digital, role-based workflow that improves efficiency, accuracy, and transparency. I provided separate access for Admin, Doctor, and Patient users to ensure secure data handling and proper authorization at every stage of the system.

Through this application, patients can register, search for doctors, view hospital departments, book appointments, and track their appointment status. The Admin module allows centralized management of doctors, patients, and appointment requests. It also includes a dashboard to monitor system activity. Doctors can update their availability and approve or reject appointments with valid reasons, reflecting real-world hospital workflows.

I built the backend using Spring Boot and Spring Data JPA, following RESTful architecture principles. CRUD operations are managed through clear API endpoints, and data is securely stored in a MySQL database using a layered architecture. Overall, this project provided me with practical experience in Java Full Stack development and serves as a solid capstone showcasing real-world application design.

### **15.2 FUTURE ENHANCEMENTS:**

While the Hospital Management System meets the current functional needs, there is significant potential for future improvements to enhance security, scalability, and usability. Integrating Spring Security with JWT can offer stronger authentication, authorization, and session management, which will better protect sensitive hospital and patient data.

Adding features like an online payment gateway would allow patients to make payments when booking appointments, which improves convenience. Implementing email and SMS notifications can help keep patients and doctors updated about appointment confirmations, rejections, and schedule changes in real time.

Other improvements include incorporating Electronic Medical Records (EMR) for the digital storage of patient medical histories and creating analytics dashboards for better decision-making. The system can also be expanded into a mobile application and deployed on cloud platforms like AWS or Azure to support scalability in large hospital settings.

## **Project Video Drive Link:**

<https://drive.google.com/drive/folders/1nmASd-ZB52DREtfL-8ihqKzLYxb8JwRI?usp=sharing>