

# 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

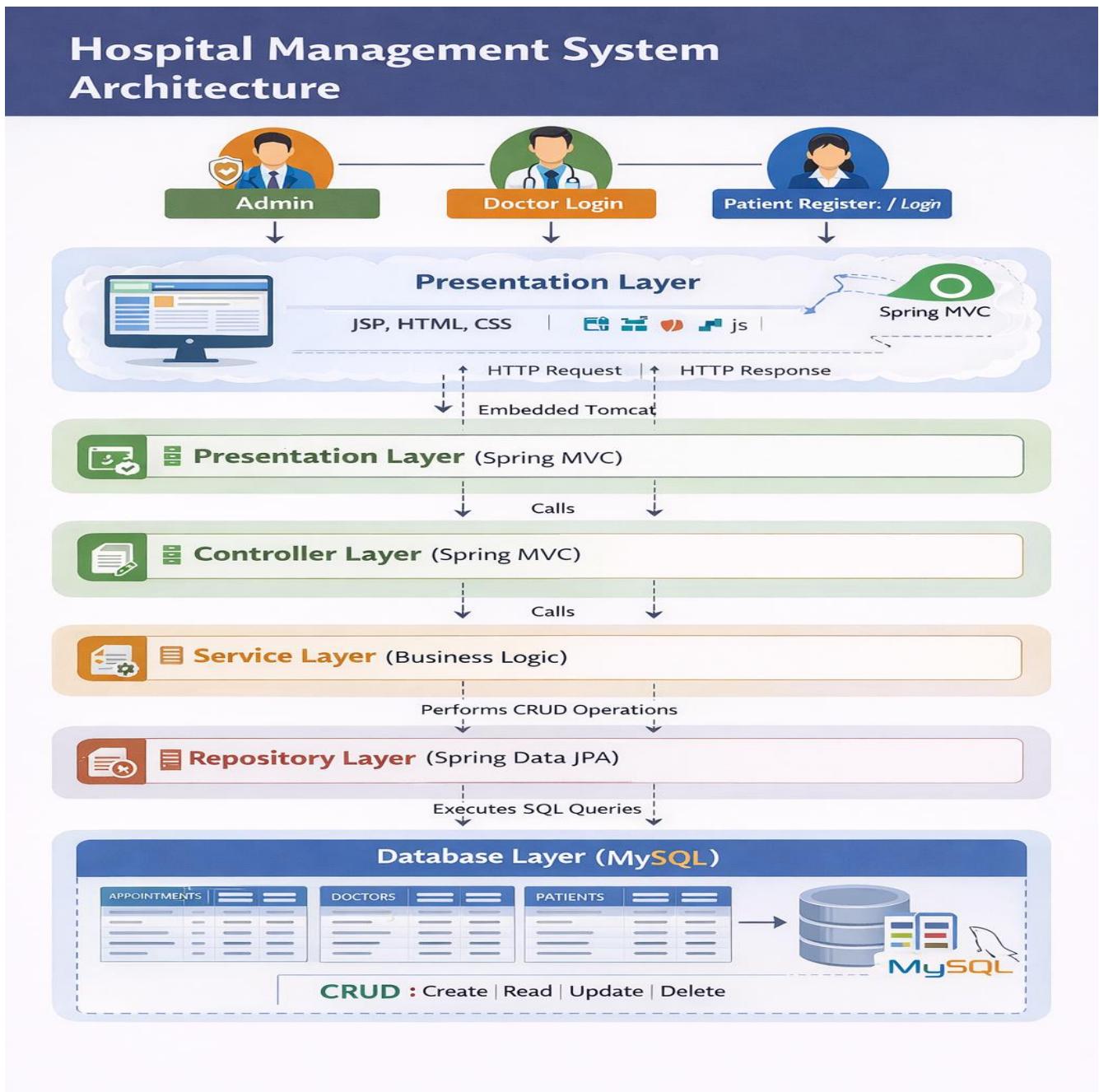
- **Processor:** Dual Core or higher
- **RAM:** Minimum 4 GB (Recommended 8 GB)
- **Hard Disk:** Minimum 5 GB free space
- **Display Resolution:** 1366 × 768 or higher
- **Internet Connection:** Required for development and deployment

### 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)

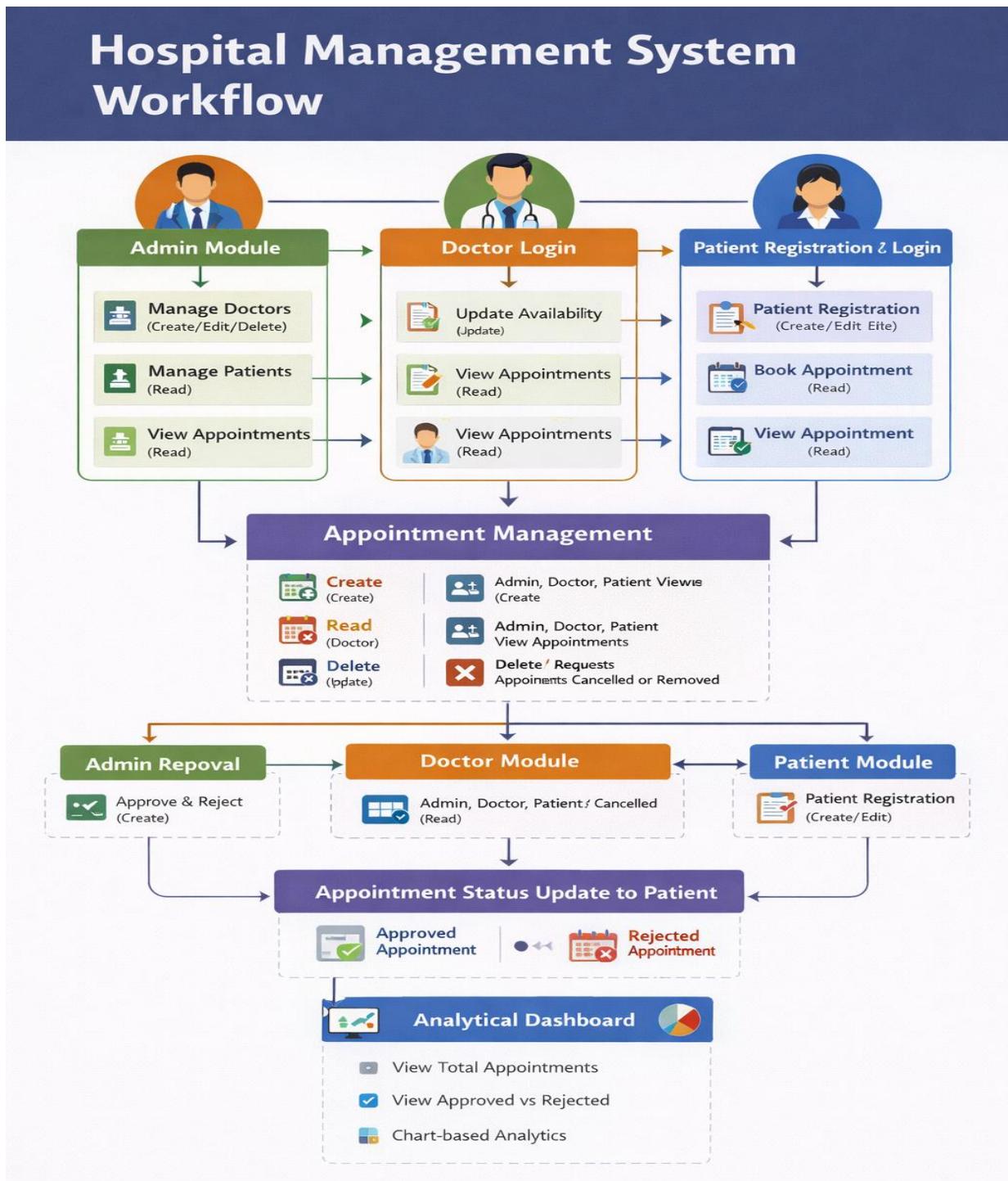
## 8.SYSTEM ARCHITECTURE

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 shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database structure under "hmsMySQL". It includes objects like "Server", "Client", "Users", "Status", "Data Ex", and "Data In".
- SQL Editor:** Displays the following SQL code:

```
1 • CREATE DATABASE hospital_db;
2 • USE hospital_db;
3 • show tables;
4 • CREATE TABLE users (
    user_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    role VARCHAR(20) NOT NULL, -- ADMIN / DOCTOR / PATIENT
    reference_id BIGINT      -- doctor_id or patient_id
);
11 • INSERT INTO users (username, password, role)
```
- Information:** Shows the "Tables\_in\_hospital\_db" table with four rows: "appointments", "doctors", "patients", and "users".
- Output:** Shows the results of the executed queries. The "Action Output" table lists the following actions and their results:

#	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.

A screenshot of a web browser displaying the 'Admin Dashboard' for a hospital management system. The title bar shows 'localhost:9095/admin-dashboard.html'. The dashboard has a blue header with 'Admin Dashboard' on the left and 'Analytics' and 'Logout' on the right. Below the header is a navigation bar with three buttons: 'Appointments' (highlighted in blue), 'Patients', and 'Doctors'. The main content area is a table titled 'Appointments' with columns: ID, Date, Slot, Patient, Doctor, Status, and Action. The table contains six rows of data. Row 3 is highlighted in green, and rows 6 and 7 are highlighted in pink. The 'Action' column for all rows displays the message 'No actions available'.

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

A screenshot of a web browser displaying the 'Admin Dashboard' for a hospital management system. The title bar shows 'localhost:9095/admin-dashboard.html'. The dashboard has a blue header with 'Admin Dashboard' on the left and 'Analytics' and 'Logout' on the right. Below the header is a navigation bar with three buttons: 'Appointments' (highlighted in blue), 'Patients' (highlighted in blue), and 'Doctors'. The main content area is a table titled 'Patients' with columns: ID, Name, Age, and Gender. The table contains six rows of data. The 'Name' column for rows 3, 4, and 5 is partially obscured by redacted text.

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	

The screenshot displays two browser windows related to a Hospital Management System.

**Top Window (Admin Dashboard):**

- Title Bar:** Admin Dashboard - Hospital M... | localhost:9095/admin-dashboard.html
- Header:** Admin Dashboard, Analytics, Logout
- Content:**
  - Section Title:** Admin Dashboard
  - Buttons:** Appointments, Patients, Doctors
  - Table:** A table listing three doctors with columns: ID, Name, Specialization, Available, and Action.
  - Buttons:** + Add New Doctor (green button) and a small minus icon.

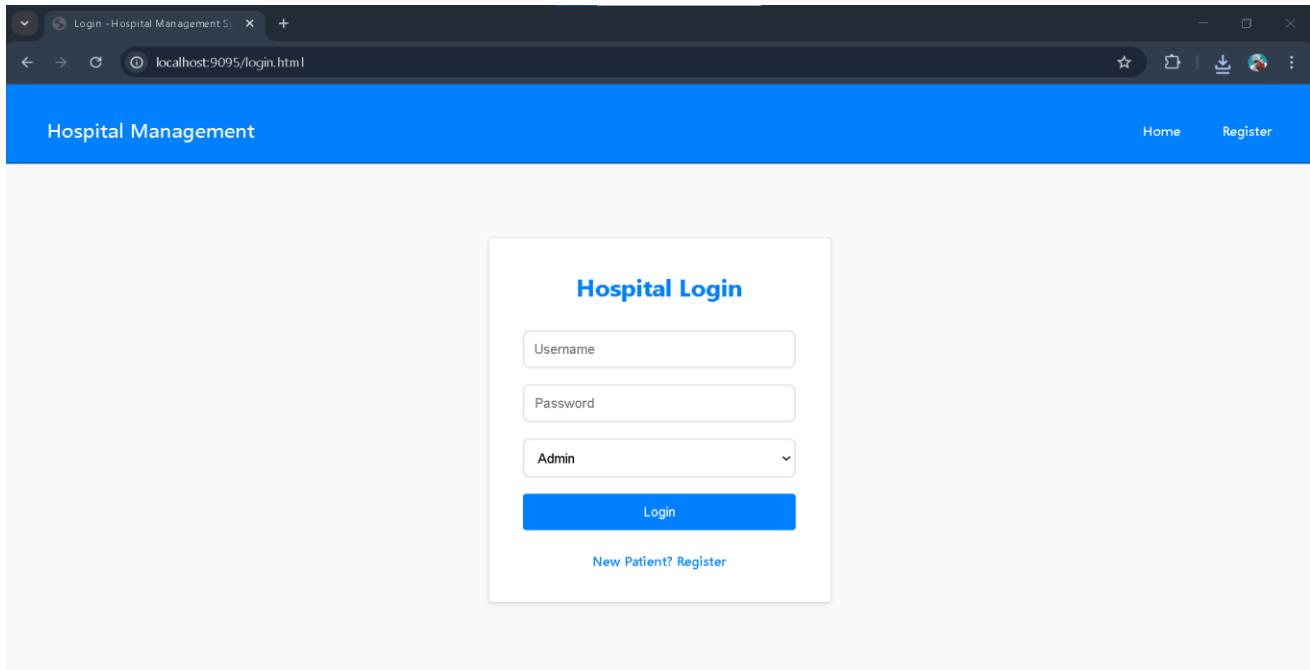
ID	Name	Specialization	Available	Action
1	Dr. Mehta	General Physician	Yes	-
2	Raghav	Ortho	Yes	-
3	A	Dermatologist	Yes	-

**Bottom Window (Add Doctor):**

- Title Bar:** Add Doctor - Hospital Management | localhost:9095/add-doctor.html
- Header:** Admin Dashboard, Back to Dashboard, Logout
- Content:**
  - Back to Dashboard** button
  - Add New Doctor** section
  - Form Fields:**
    - Doctor ID: Will be: 4 (Note: ID will be automatically generated after the current number of doctors)
    - Doctor Name \*: Enter doctor name
    - Specialization \*: Enter specialization (e.g., Cardiology, Neurology)
    - Availability: Available dropdown
  - Buttons:** Add Doctor (blue button)

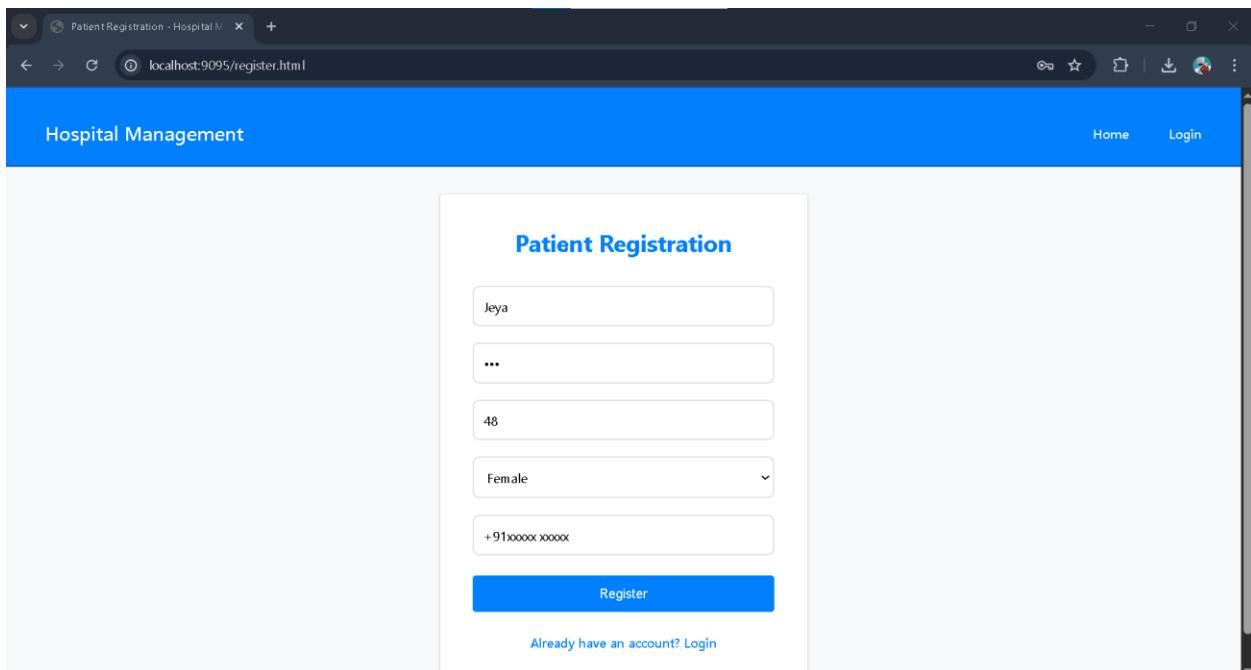
## 12.2 Login Module:

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.



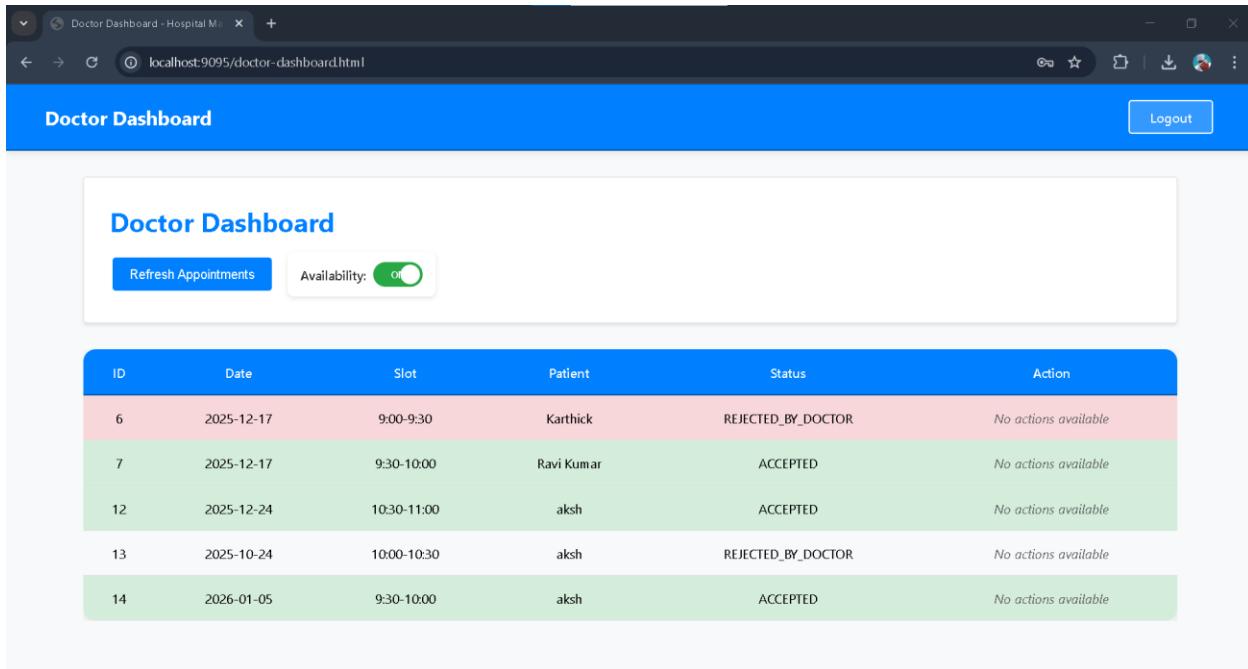
### 12.3 Register Module:

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.



## 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.

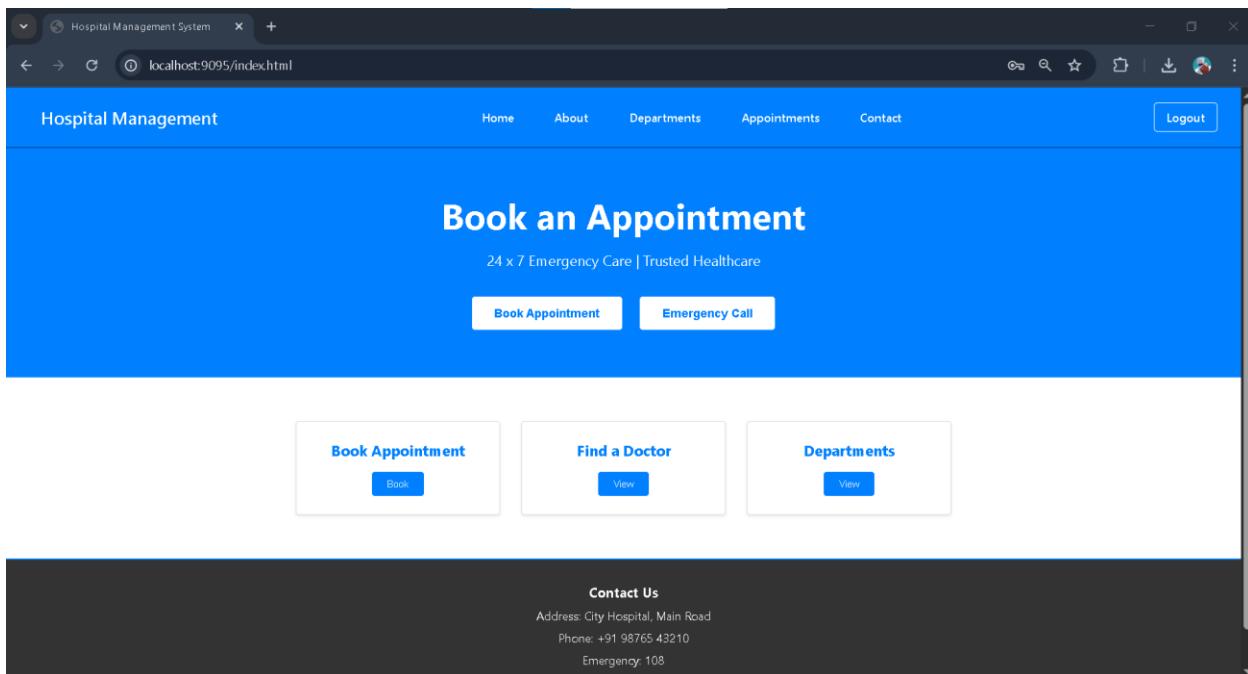


A screenshot of a web browser displaying the 'Doctor Dashboard' at localhost:9095/doctor-dashboard.html. The dashboard has a blue header with the title 'Doctor Dashboard' and a 'Logout' button. Below the header is a search bar with 'Refresh Appointments' and 'Availability: On' buttons. The main area contains a table with columns: ID, Date, Slot, Patient, Status, and Action. The table shows five rows of appointment data. Rows 6 and 13 are pink (REJECTED\_BY\_DOCTOR), while others are green (ACCEPTED). Each row includes a 'No actions available' message in the Action column.

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

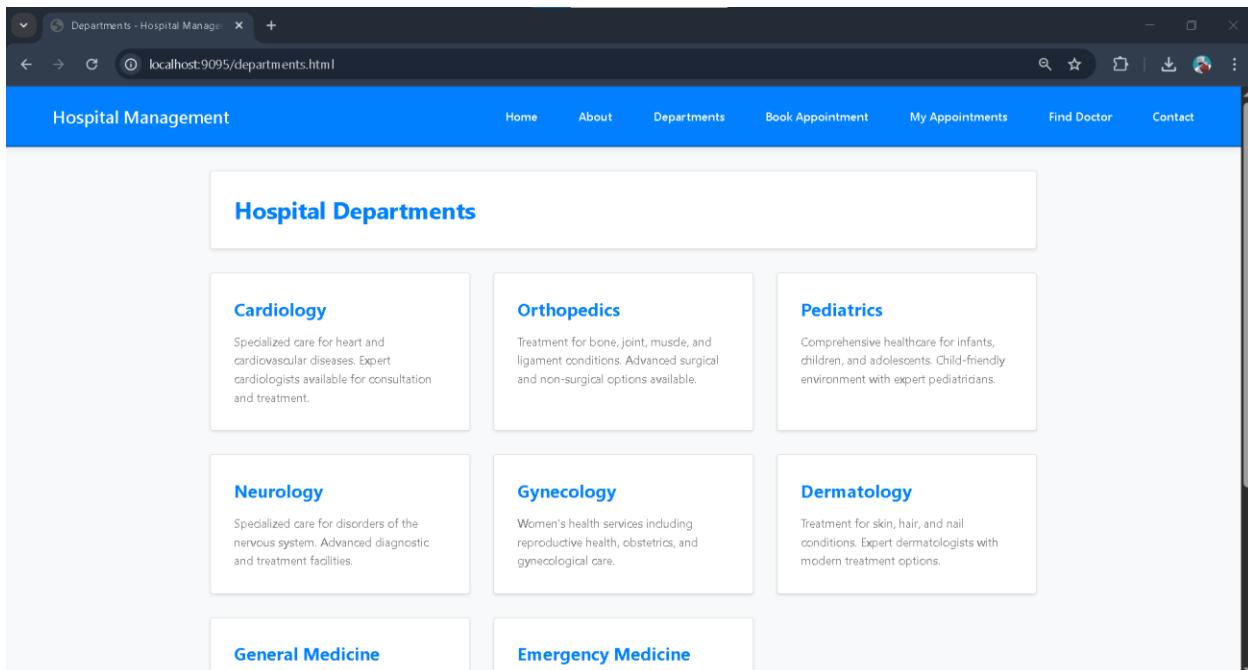
## 12.5 Patient Home & Appointment Booking Module

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.

A screenshot of a web browser showing the 'Book Appointment' page of a hospital management system. The URL in the address bar is 'localhost:9095/appointment.html'. The page has a blue header with the text 'Hospital Management' and navigation links for 'Home', 'About', 'Departments', 'Appointments', and 'My Appointments'. The main content area is titled 'Book Appointment' and contains the following form fields:

- Input field with value '5'
- Input field labeled 'Doctor ID'
- Input field labeled 'Age'
- Dropdown menu labeled '-- Select Gender --'
- Input field with placeholder 'mm / dd / yyyy' and a calendar icon
- Dropdown menu labeled '-- Select Time Slot --'
- Dropdown menu labeled 'Normal'
- A large 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.

The screenshot shows a web browser window titled "Appointments List - Hospital M..." with the URL "localhost:9095/appointments-list.html". The page has a blue header bar with the text "Hospital Management" and navigation links for "Home", "About", "Departments", "Book Appointment", and "My Appointments". Below the header is a section titled "My Appointments" with a "Load Appointments" button. A table displays appointment details:

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.

The screenshot shows a web browser window titled "About Us - Hospital Management" with the URL "localhost:9095/about.html". The page has a blue header bar with the text "Hospital Management" and navigation links for "Home", "About", "Departments", "Appointments", and "Contact", along with a "Logout" button. The main content area features a section titled "About Our Hospital" with the tagline "Committed to providing exceptional healthcare services with compassion and excellence". Below this are three sections: "Welcome to Our Hospital", "Our Mission", and "Our Vision".

**Welcome to Our Hospital**  
Our hospital is a leading healthcare institution dedicated to providing comprehensive medical care to our community. With state-of-the-art facilities and a team of experienced healthcare professionals, we strive to deliver the highest quality medical services with compassion and excellence.  
Established with a vision to make quality healthcare accessible to everyone, we have been serving our community for years, building trust and delivering exceptional patient care. Our commitment to excellence has made us a preferred choice for medical care in the region.

**Our Mission**  
To provide accessible, high-quality healthcare services that improve the health and well-being of our community. We are committed to delivering patient-centered care with integrity, compassion, and respect for all individuals.

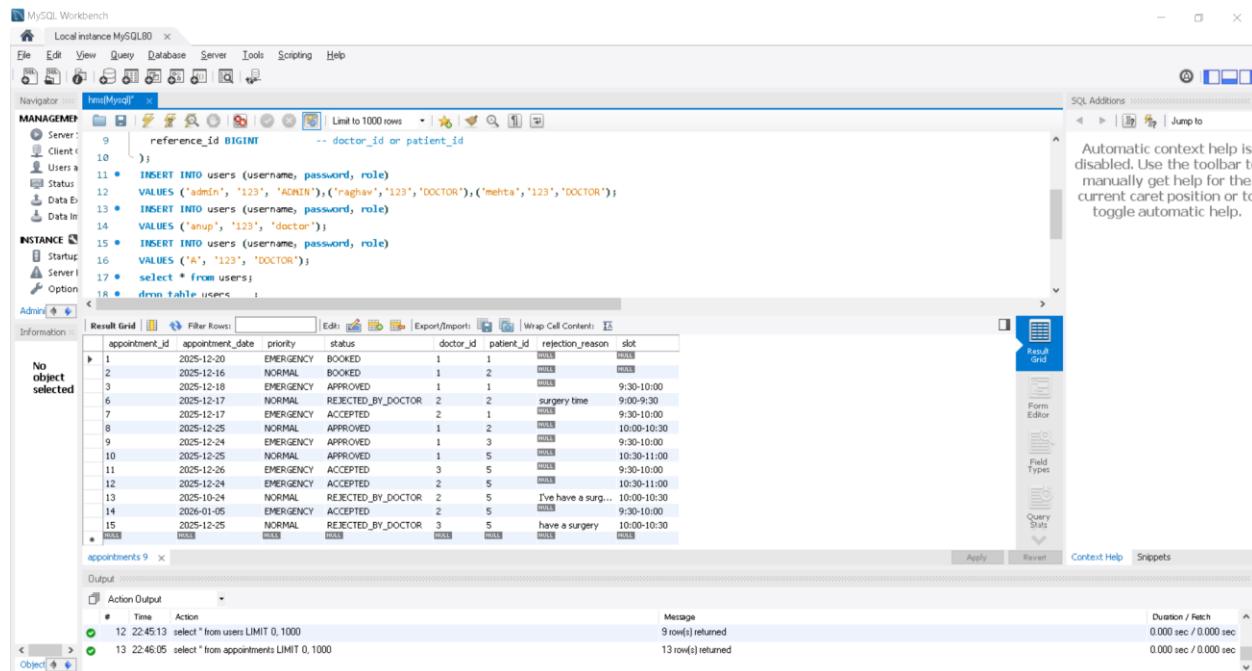
**Our Vision**  
To be recognized as a premier healthcare institution that sets the standard for excellence in medical care, innovation, and patient satisfaction. We

## 13.DATABASE TABLES

### 13.1 Users Table:

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

## 13.2 Appointments Table:



The screenshot shows the MySQL Workbench interface with the 'appointments' table selected in the results grid. The table has columns: appointment\_id, appointment\_date, priority, status, doctor\_id, patient\_id, and rejection\_reason. The data is as follows:

appointment_id	appointment_date	priority	status	doctor_id	patient_id	rejection_reason
1	2025-12-20	EMERGENCY	BOOKED	1	1	
2	2025-12-16	NORMAL	BOOKED	1	2	
3	2025-12-18	EMERGENCY	APPROVED	1	1	
4	2025-12-17	NORMAL	REJECTED_BY_DOCTOR	2	2	surgery time
7	2025-12-17	EMERGENCY	ACCEPTED	2	1	
9	2025-12-25	NORMAL	APPROVED	1	2	
10	2025-12-25	NORMAL	APPROVED	1	3	
11	2025-12-26	EMERGENCY	ACCEPTED	3	5	
12	2025-12-24	EMERGENCY	ACCEPTED	2	5	
13	2025-10-24	NORMAL	REJECTED_BY_DOCTOR	2	5	I've have a surg...
14	2026-01-05	EMERGENCY	ACCEPTED	2	5	
15	2025-12-25	NORMAL	REJECTED_BY_DOCTOR	3	5	have a surgery
16	2025-12-25	NORMAL	REJECTED_BY_DOCTOR	3	5	

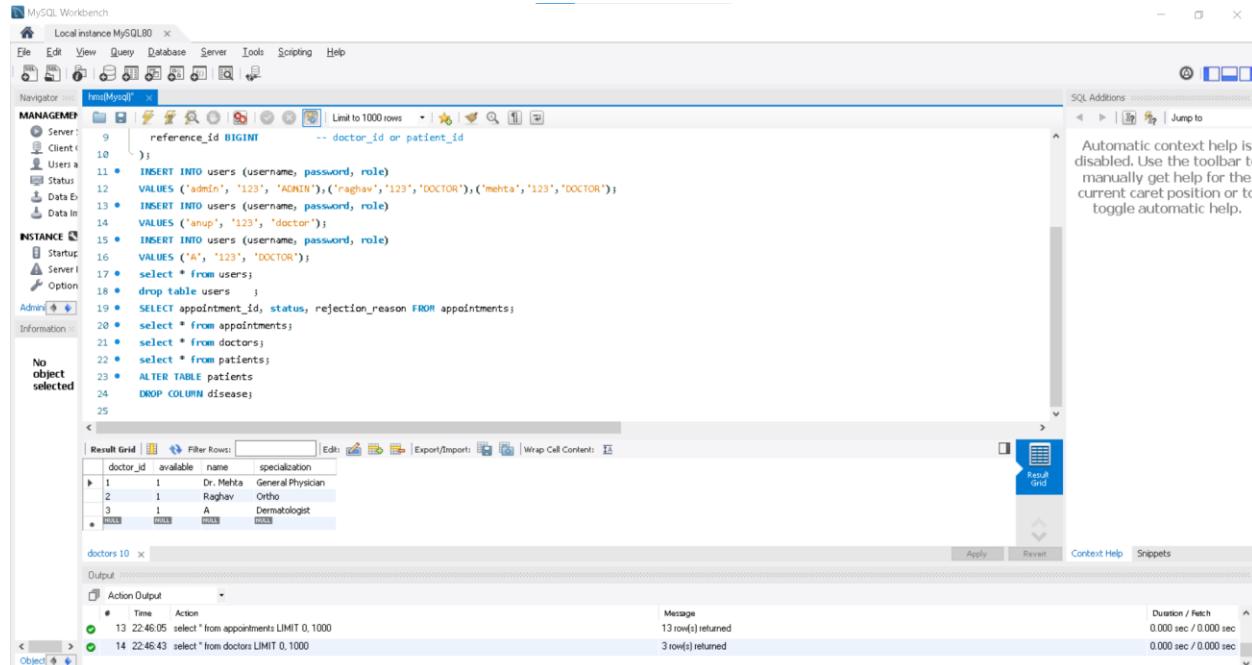
The SQL editor contains the following code:

```

MANAGEMENT
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;

```

## 13.3 Doctors Table:



The screenshot shows the MySQL Workbench interface with the 'doctors' table selected in the results grid. The table has columns: doctor\_id, available, name, and specialization. The data is as follows:

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

The SQL editor contains the following code:

```

MANAGEMENT
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;
20  • select * from appointments;
21  • select * from doctors;
22  • select * from patients;
23  • ALTER TABLE patients
24  • DROP COLUMN disease;
25

```

## 13.4 Patients Table:

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Displays the SQL code used to create the 'patients' table and insert data. The table structure includes columns: patient\_id (INT), age (TINYINT), gender (CHAR(1)), name (VARCHAR(50)), phone (VARCHAR(15)), and disease (VARCHAR(50)).
- Result Grid:** Shows the data inserted into the 'patients' table. There are 6 rows of data.
- Output Panel:** Displays the execution log for the queries run. It shows two successful SELECT statements with their execution time and message.

patient_id	age	gender	name	phone	disease
1	30	Male	Ravi Kumar	9876543210	NULL
2	19	Male	Karthik	7565696331	NULL
3	0		Mondin	NULL	NULL
4	0		asid	NULL	NULL
5	20	Female	aksh	NULL	NULL
6	0		IQ	NULL	NULL

```
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)
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;
```

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

## 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.

The screenshot shows the Spring Tools for Eclipse interface. On the left, the Package Explorer displays the project structure under 'hospital-management-system'. The central area shows a code editor with a snippet of CSS for a navbar. Below it is a terminal window displaying the application's startup logs. The logs show the application starting up, initializing Spring Data repositories, and connecting to a MySQL database using HikariCP.

```

2025-12-18T22:49:43.492+05:30 INFO 13220 --- [ restartedMain] c.h.HospitalManagementSystemApplication : Starting HospitalManagementSystemApplication using Java 17.0
2025-12-18T22:49:43.492+05:30 INFO 13220 --- [ restartedMain] c.h.HospitalManagementSystemApplication : No active profile set, falling back to 1 default profile: "dev"
2025-12-18T22:49:43.527+05:30 INFO 13220 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' or 'spring.devtools.restart.enabled=true' for additional web related logging consider setting the 'logLevel' property for more detailed logging
2025-12-18T22:49:43.527+05:30 INFO 13220 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Bootstrapping Spring Data JPA repositories in DEFAULT mode
2025-12-18T22:49:43.527+05:30 INFO 13220 --- [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 99 ms. Found 4 JPA
2025-12-18T22:49:44.203+05:30 INFO 13220 --- [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Tomcat initialized with port 9095 (http)
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Starting service [Tomcat]
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Starting Servlet engine: [Apache Tomcat/10.1.49]
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : StandardService
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] o.a.catalina.core.StandardEngine : Initialization context: initialization completed in 1295 ms
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] o.h.b.dialect.Dialect : HHH000412: Hibernate Core version 6.6.36.Final
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] o.h.b.Version : HHH000266: Second-level cache disabled
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] o.h.b.Version : No LoadTimeWeaver setup; ignoring JPA class transformer
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] HikariPool-1 : HikariPool-1 - Starting...
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] HikariPool-1 : HikariPool-1 - Added connection com.mysql.cj.jdbc.Connection
2025-12-18T22:49:44.319+05:30 INFO 13220 --- [ restartedMain] HikariPool-1 : HikariPool-1 - Start completed.
2025-12-18T22:49:45.563+05:30 WARN 13220 --- [ restartedMain] org.hibernate.orm.deprecation : HHH9000025: MySQLDialect does not need to be specified exp:
2025-12-18T22:49:45.581+05:30 INFO 13220 --- [ restartedMain] org.hibernate.orm.connections.pooling : HHH9000026: MySQLDialect has been deprecated; use org.hibernate.dialect.MySQLDialect instead
2025-12-18T22:49:45.581+05:30 INFO 13220 --- [ restartedMain] org.hibernate.orm.connections.pooling : HHH10001005: Database info:

```

## 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.

The screenshot shows the Postman application interface. A collection named 'Patients' is selected. A GET request is made to 'http://localhost:9095/patients'. The response status is 200 OK, and the response body is displayed as JSON, showing a list of patient records.

```

[{"patientId": 1, "name": "Ravi Kumar", "age": 30, "gender": "Male", "phone": "+9176543210", "disease": null}, {"patientId": 2, "name": "Karthick", "age": 19, "gender": "Male", "phone": "+9165696331", "disease": null}, {"patientId": 3, "name": "Monshin", "age": 0, "gender": "", "phone": ""}]

```

## **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>