Hospital Management System (HMS)

App Dev I Final Project Report

Rounak Mukhopadhyay
Roll No: 22f1000876
22f1000876@ds.study.iitm.ac.in

December 1, 2025

Contents

1 Introduction

- **Objective of the Project:** The goal of this project is to design and develop a role-based Hospital Management System (HMS) that supports secure login, doctor availability, appointment management, and treatment history through a unified web interface.
- **Motivation:** Many mid- and small-scale hospitals rely on manual scheduling and paper-based patient history, leading to double bookings and information loss. HMS provides centralized appointment allocation and reliable medical record tracking.
- **Web Applications in Healthcare:** Role-based web systems allow patients, doctors, and hospital staff to access the same database through permission-controlled dashboards, improving transparency and reducing administrative overhead.

2 System Overview

2.1 Features Implemented

- **Role-Based Access Control:**
  - **Admin:** Manages doctors, patients, and all appointments. Dashboard shown in Figure 7.
  - **Doctor:** Views upcoming appointments, manages availability, and records treatment (Figure 5).
  - **Patient:** Registers, books appointments, views history (Figure 3).
- **Authentication and Redirection:** A unified login page routes users to their respective dashboards based on role (Figure 2).
- **Admin Functionalities:** Add/update/blacklist doctors, manage patients, and view *all appointments* (Figure 8).
- **Doctor Functionalities:** View appointments and record treatment using a structured form (Figure 6).
- **Patient Functionalities:** Find doctors and book appointments within allowed availability windows (Figure 4).

2.2 Technology Stack

Technology	Role in Project
Flask	Backend routing and business logic
SQLite	Relational database for persistent storage
Jinja2	Server-side template rendering
Bootstrap	Responsive page layout and styling
Werkzeug Security	Password hashing and authentication safety

Chart.js	Appointment status visualization on admin dashboard

3 Architecture

3.1 Directory Structure

```
hospital_app/
 app.py
 models/
    models.py
 controllers/
    routes.py
 templates/
    base.html
    login.html
    dashboard_admin.html
    dashboard_doctor.html
    dashboard_patient.html
    admin_appointments.html
    patient_book_appointment.html
    doctor_treatment_form.html
 instance/hospital.db
 static/ (CSS/JS)
```

3.2 ER Diagram

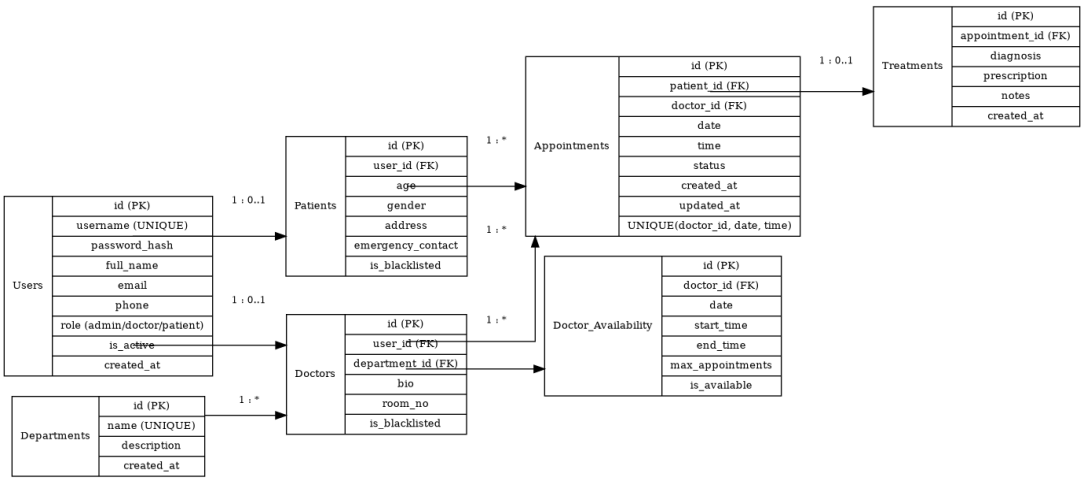

Figure 1: Entity Relationship Diagram of the Hospital Management System.

4 User Interface Screens

This section highlights the most important user flows of the Hospital Management System (HMS). Each interface has been designed using HTML, Bootstrap, and Jinja2 to provide a clean and role-specific experience.

4.1 Login and Registration

The system begins with a unified login interface for all roles. Patients can also register themselves from this screen.

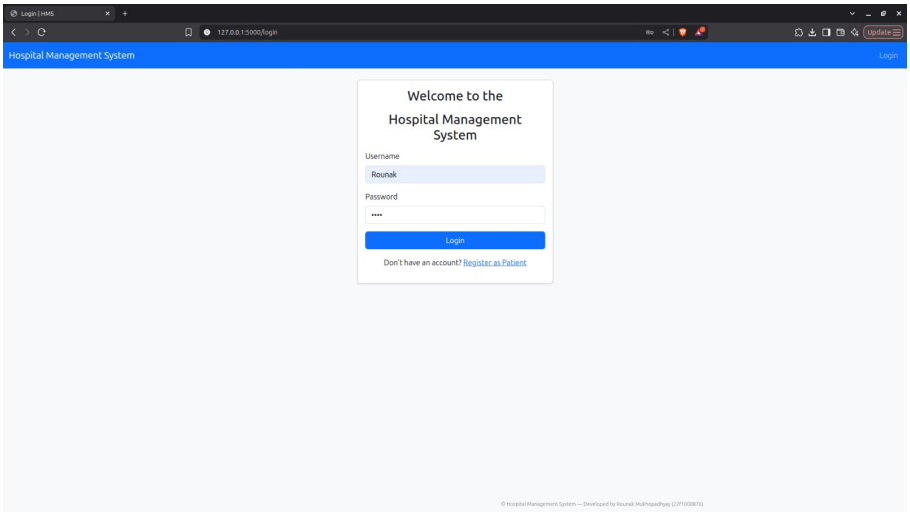

Figure 2: Unified login page used by Admin, Doctor, and Patient.

4.2 Patient Dashboard and Appointment Booking

Once logged in, patients are redirected to a dashboard showing all medical departments and doctor availability for the next 7 days (Figure 3). Patients can search doctors by specialization and book appointments through a guided date–time selection interface (Figure 4).

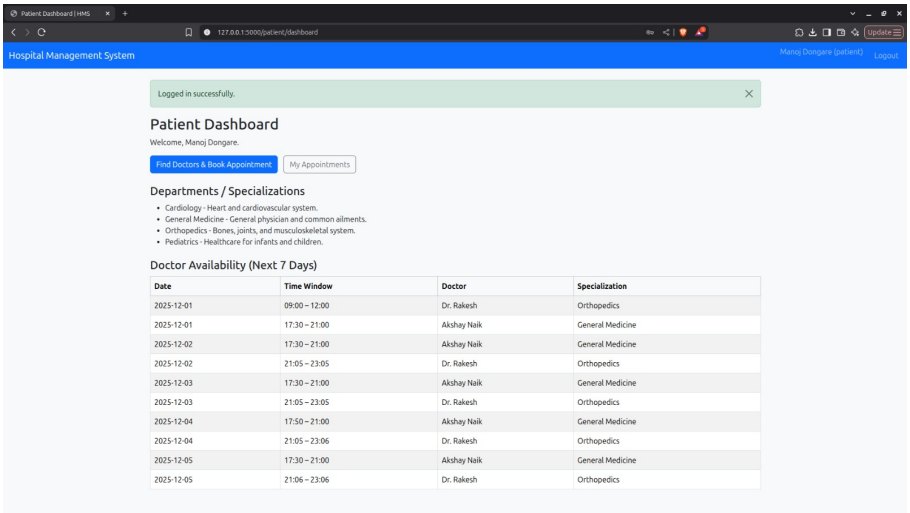

Figure 3: Patient dashboard displaying departments and weekly doctor availability.

4.3 Doctor Dashboard and Treatment Workflow

Doctors get a dedicated dashboard summarizing upcoming and completed appointments (Figure 5). For each completed visit, the doctor can open a treatment form to record diagnosis, prescription, and notes (Figure 6).

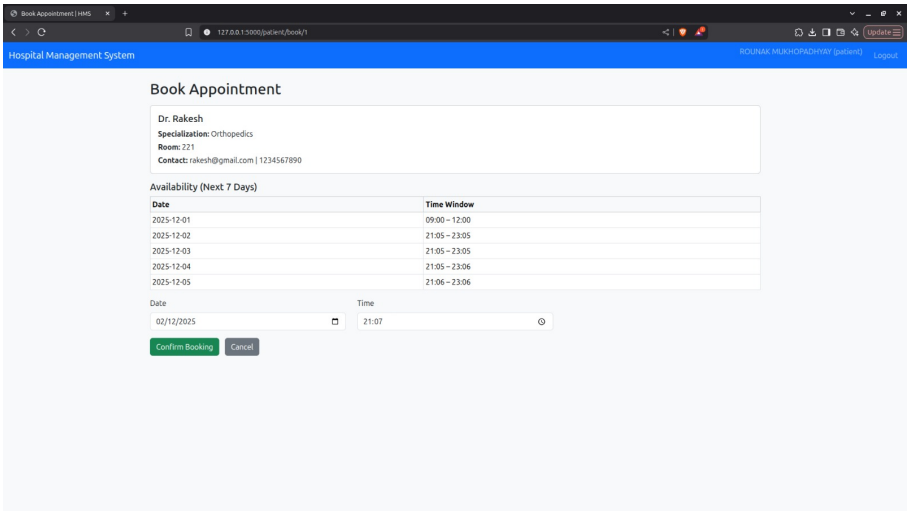

Figure 4: Appointment booking screen showing doctor availability and time-slot validation.

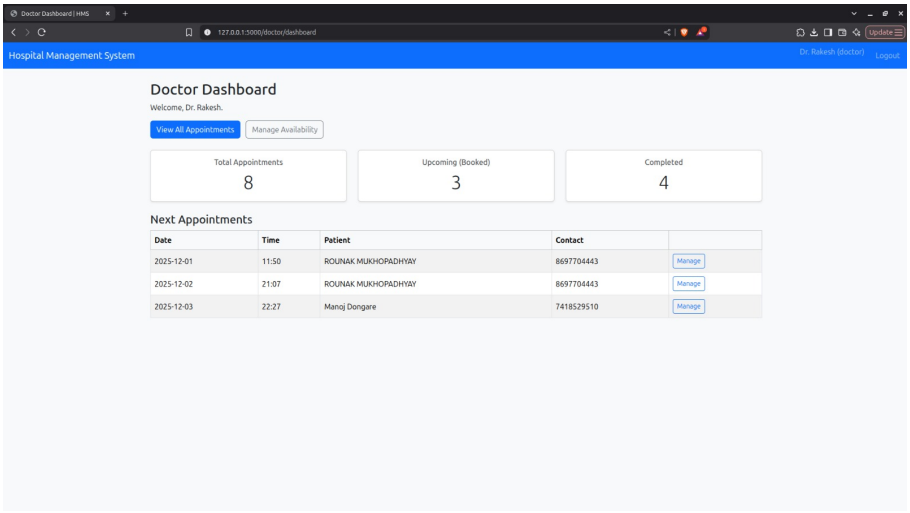

Figure 5: Doctor dashboard showing total, upcoming, and completed appointments.

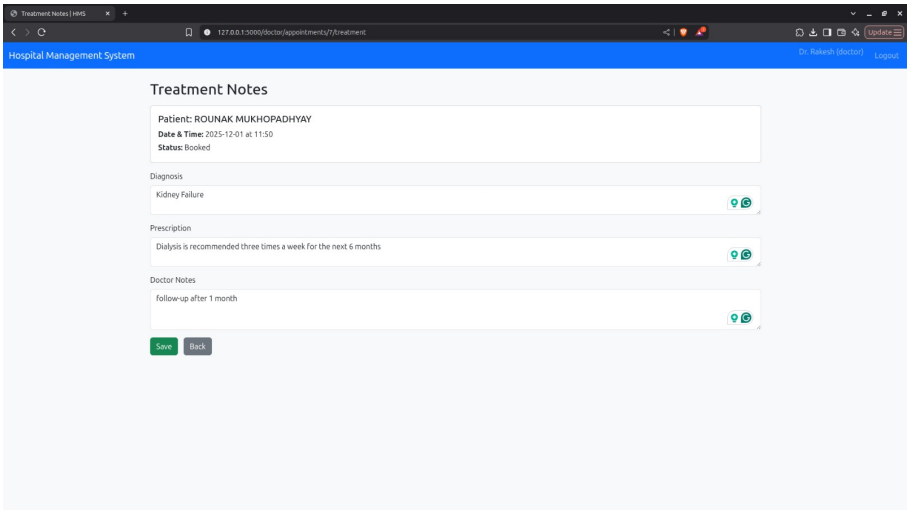

Figure 6: Treatment form for recording diagnosis, prescription, and doctor notes.

4.4 Admin Dashboard and Hospital Operations

Admins can monitor and manage hospital operations from a centralized panel. The dashboard (Figure 7) shows total counts of doctors, patients, and appointments, along with a pie-chart visualization of appointment statuses. Admins can also view and filter all appointments through the dedicated panel (Figure 8).

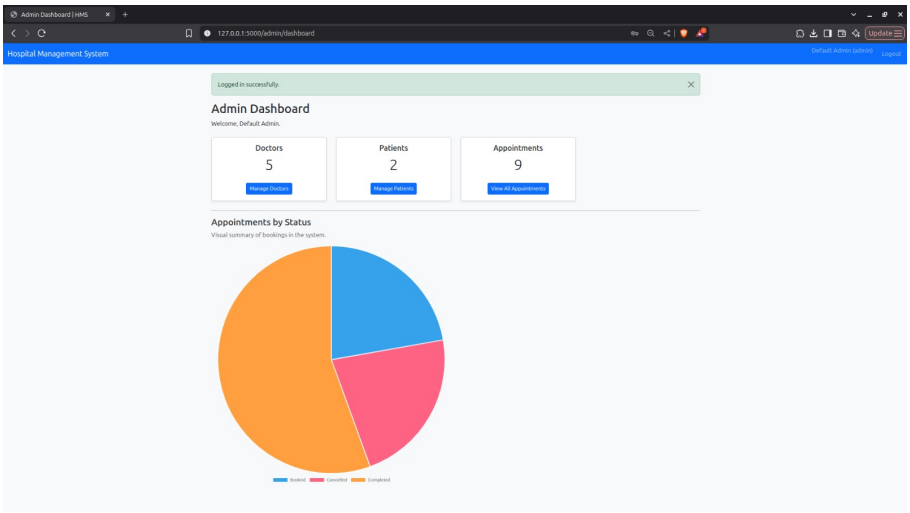

Figure 7: Admin dashboard with summary counts and appointment status visualization.

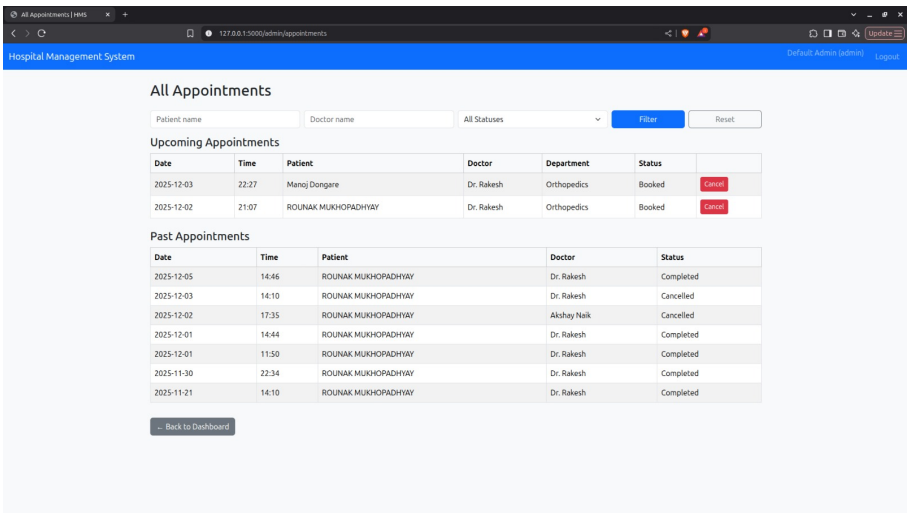

Figure 8: Admin view of all appointments with filtering by patient, doctor, and status.

5 AI / LLM Assistance Declaration

- ChatGPT was used only for brainstorming architecture ideas, extracting code templates, debugging, and drafting documentation.
- All code, logic, and UI implementation were manually written, tested, and modified by the student.
- AI assistance did not access or interact with private institutional data or user credentials.

6 Contribution Breakdown

Component / Task	Weight	Description
Flask Backend Development	25%	Routing, authentication, database communication
Admin Functionalities	15%	Manage doctors, patients, and appointments
Appointment & Availability Logic	20%	Slot validation and double-booking prevention
Treatment History Implementation	10%	Diagnosis, prescriptions, and notes per visit
Frontend (HTML/CSS/Jinja2)	20%	Interface design for all three roles
Analytics / API / Visualization	5%	Chart.js graph + `/api/stats` endpoint
AI Assistance	5%	Debugging and documentation support

7 Challenges Faced

- Implementing availability windows and validating appointment time boundaries.
- Ensuring the doctor cannot be double-booked via uniqueness constraints and business logic.
- Maintaining clean UI while supporting three distinct user flows with minimal code duplication.
- Keeping database schema simple yet extensible for future HMS expansion.

8 Conclusion and Future Scope

- A complete Hospital Management System was developed with role-based access and persistent treatment history.
- Future enhancements include laboratory report uploads, mobile app support, SMS/email reminders, and pharmacy/lab workflow modules.

9 References

- Flask: `https://flask.palletsprojects.com/`
- SQLite: `https://www.sqlite.org/docs.html`
- Bootstrap: `https://getbootstrap.com/`
- Chart.js: `https://www.chartjs.org/`
- StackOverflow and GitHub Discussions