Hospital Management System (HMS) - V1

Project Report

Student Details

Name:	Lakshman Raj B
Program:	BS Degree in Data Science and Applications
Level:	Diploma Level
Roll Number:	21F1000630
Video link:	MAD-I demo link

App Dev I — Project Submission
September 2025

AI/LLM Declaration

I acknowledge that I used AI/LLM assistance (ChatGPT by OpenAI) during the development of this project. The estimated extent of usage is approximately **30%** of the entire project.

The AI assistance was used only for the following purposes:

- Primarily in the frontend (HTML, Bootstrap) and a few parts of the backend (Flask, SQL queries).
- **Understanding documentation** for technologies such as Flask login and flask security application context.

No part of the project was autogenerated end-to-end by the AI except db seeding. All core logic, features, design decisions, database schema, and integrations were implemented by me. The AI was strictly used as a supportive tool for pair programming, debugging, clarification, and improving code quality.

Introduction

This report explains the backend design and working of my **Hospital Management System (HMS)** project. The goal of this system is to make hospital activities simple and digital, so the users can manage appointments, doctors, patients, and medical records easily.

The system has three main users:

- **Admin** – manages doctors, patients, departments, and all appointments.
- **Doctor** – checks appointments, marks visits completed, adds diagnosis and prescription, and adds availability for next 7 days.
- **Patient** – registers, books appointments, views history, reschedules or cancels.

The complete backend is developed using Flask and SQLite. The system is simple, modular, and easy to maintain.

System Overview

Components

- **app.py** – Main file which runs the app, loads routes, and initializes the database.
- **db.py** – Creates SQLite database automatically from schema.sql and inserts default admin.
- **security.py** – Role checking, login protection, and session restrictions.
- **admin_routes.py** – All admin pages and logic.
- **doctor_routes.py** – Doctor dashboard and visit update logic.
- **patient_routes.py** – Registration, booking, and appointment management.
- **Templates folder** – HTML + Bootstrap pages with Jinja2.

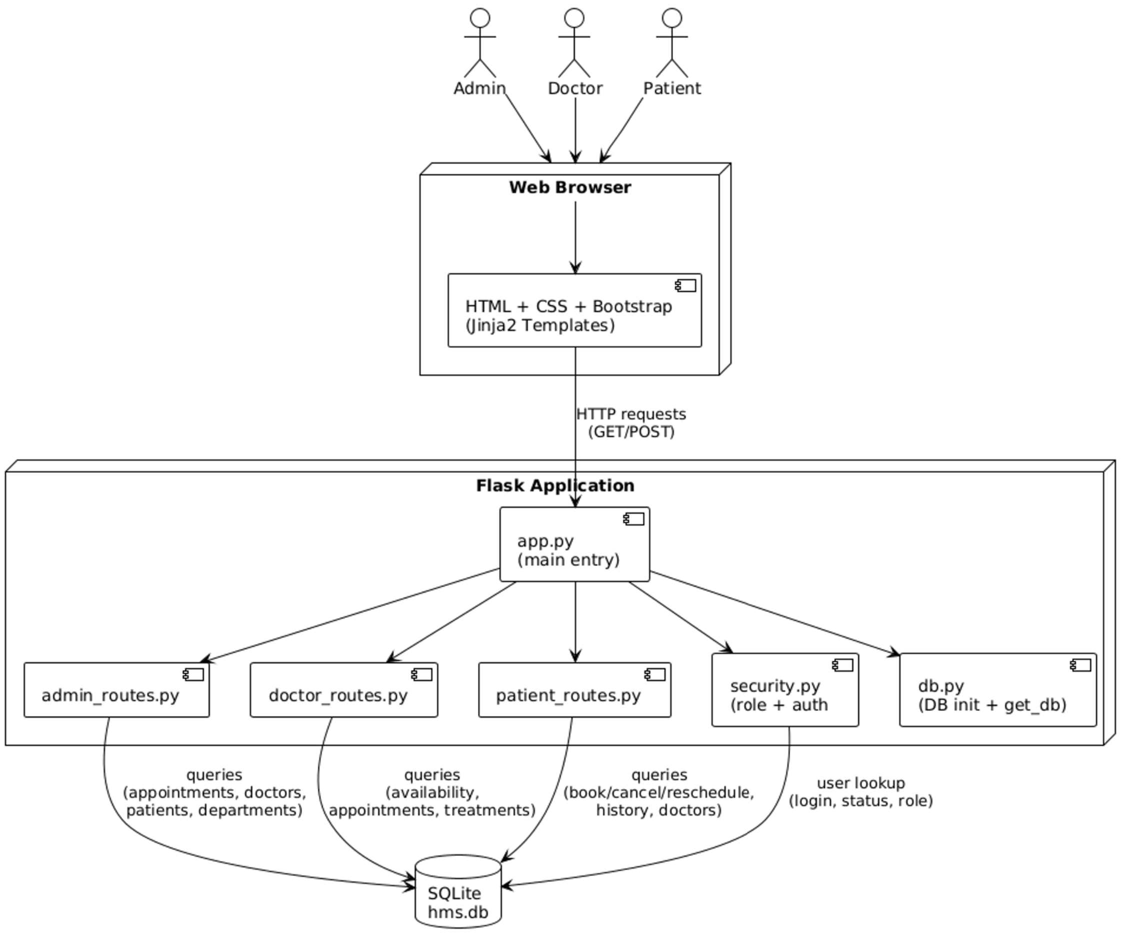

Figure 1: High Level Architecture of the Hospital Management System

High Level Design

- User logs in using email and password.
- System checks user role (Admin/Doctor/Patient).
- User is redirected to their dashboard.
- Admin can manage all data.
- Doctor can handle appointments and add treatments.
- Patient can book or modify appointments.

Database Design

The database is created automatically from the file `schema.sql`. No table is created manually. The system creates all tables on first run. The main tables in the database are:

Table Name	**Description**
users	Stores login information like email, password hash, role, and status
doctor_profiles	Doctor details such as name, specialization, phone and department
patient_profiles	Patient details like name, age, gender, phone and address
departments	Stores department names and descriptions
doctor_availability	Doctor available slots for the next 7 days
appointments	Appointment requests including date, time, doctor, patient and status
treatments	Stores diagnosis and prescription for completed appointments

Key Database Rules

- One user can be only one type: admin, doctor or patient.
- Each doctor has exactly one doctor_profile.
- Each patient has exactly one patient_profile.
- A doctor cannot have two appointments on the same date and time (UNIQUE rule).
- Foreign keys are used for linking tables.
- Every completed appointment will have one treatment entry.

ER Diagram

Below figure shows the Entity–Relationship diagram of the database. It represents the connection between users, doctors, patients, appointments, and treatments.

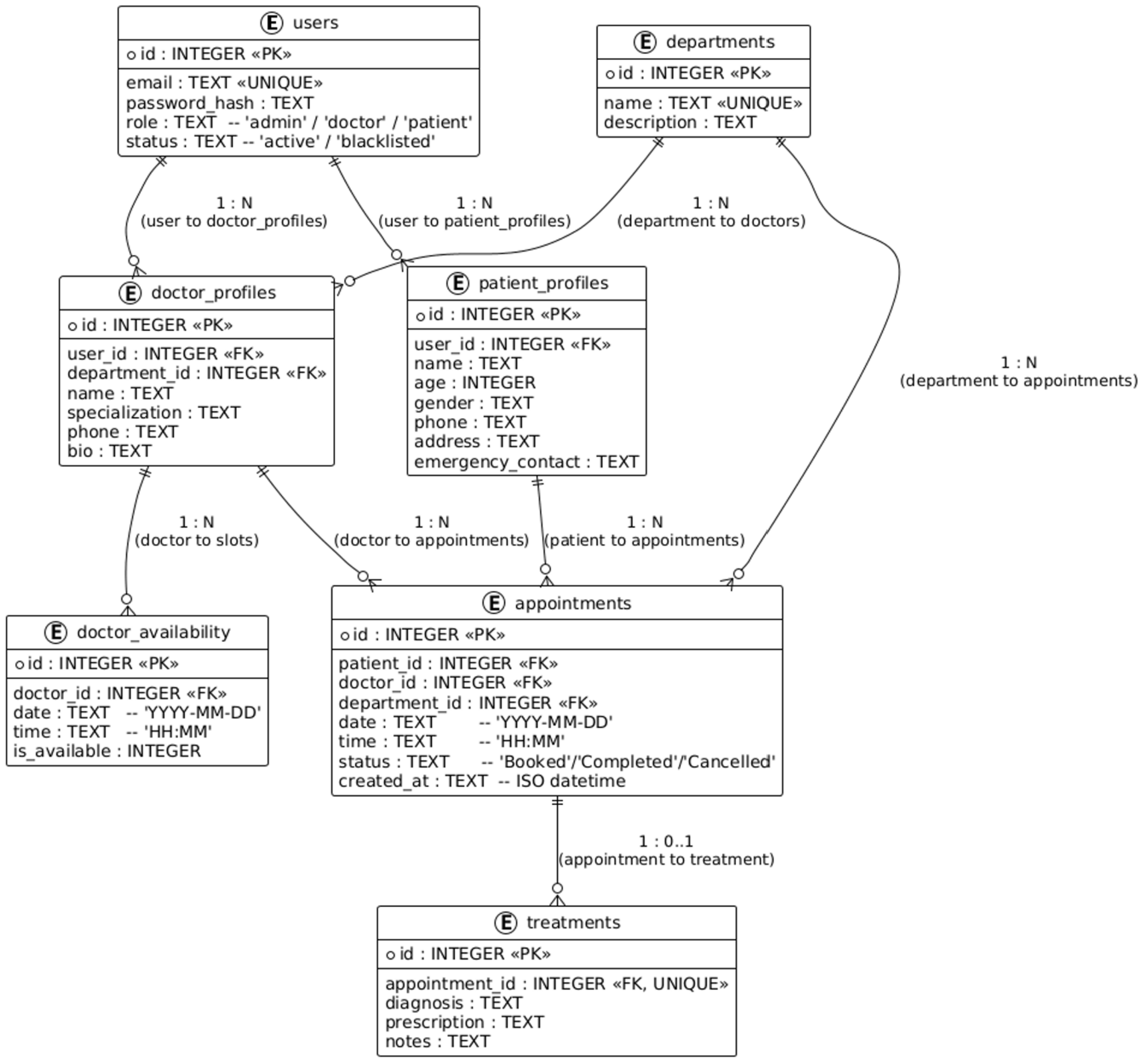

Figure 2: ER Diagram of Hospital Management System

Entities and Relationships

Entity	**Relationship Description**
Users	Linked to doctor_profiles or patient_profiles based on role
Doctor_profiles	One doctor_profile belongs to exactly one user
Patient_profiles	One patient_profile belongs to exactly one user
Departments	One department can have many doctors
Appointments	Connects doctor and patient with date and time
Treatments	One treatment is linked to exactly one appointment
Doctor_availability	Shows available slots for each doctor

Role-wise Features

Admin Features

Feature	**Description**
Dashboard stats	Shows total doctors, patients and appointments.
Manage doctors	Add, edit and blacklist/activate doctors.
Manage patients	View and edit patient profiles.
View appointments	Filter appointments by status and see details.

Doctor Features

Feature	**Description**
Doctor dashboard	Shows upcoming appointments for next 7 days.
Update appointments	Mark as Completed/Cancelled and add diagnosis and prescription.
Set availability	Add available time slots for next 7 days.
View patient history	See past completed visits with treatment details.

Patient Features

Feature	**Description**
Registration and login	Create patient account and login securely.
Dashboard	Shows departments, upcoming and past appointments.
Find doctors	Search by name, specialization or department.
Book / cancel / reschedule	Manage appointments with doctor availability validation.
View treatment history	See diagnosis and prescriptions for completed visits.

Conclusion

This Hospital Management System (HMS) implements a basic but complete workflow for admin, doctor and patient roles. The system is built with Flask and SQLite, and all tables are created programmatically. Core features like role-based login, appointment booking, availability management and treatment history are implemented.