# Hospital Management System

# App Dev-1 Project Report

## 1. Student Details

**Name:** Mohit Kumar Rai
**Roll Number:** 24F3004415
**Email:**24f3004415@ds.study.iitm.ac.in

**About Me:** I am Mohit, an undergraduate student pursuing a BS in Data Science from IIT Madras. I am passionate about building practical software solutions and enjoy applying my knowledge of Python, SQL, and web development to real-world projects.

---

## 2. Project Details

### Project Title: Hospital Management System

### Problem Statement:

Hospitals need efficient systems to manage patients, doctors, appointments, and treatments. Currently, many hospitals use manual registers or disconnected software, which makes it difficult to manage records, avoid scheduling conflicts, and track patient history.

### Approach:

The app was built using Flask as the backend framework with a modular structure. My approach for developing the Hospital Management System (Matrix) began with clearly identifying the core requirements such as user roles, departments, appointments, and authentication. I designed the database schema first, ensuring proper relationships between users, doctors, patients, and appointments. After setting up the models in Flask-SQLAlchemy, I built modular routes and templates to handle each function systematically. I focused on clean UI design using Bootstrap and ensured smooth workflow integration between all modules. Throughout the development, I tested each feature step-by-step to maintain accuracy, reliability, and user-friendly performance.

---

## 3. AI/LLM Declaration

I used **ChatGPT (GPT-5)** to assist in writing SQLAlchemy model definitions, creating bootstrap templates and for debugging purpose.

The extent of AI/LLM usage is around **10–15%**, limited to **code suggestions and documentation formatting**.

All final implementation logic, debugging, and integration were done manually.

---

# 4. Technologies and Frameworks Used

| Technology / Library | Purpose |
|---|---|
| **Flask** | Core backend web framework |
| **SQLAlchemy** | Object Relational Mapper for SQLite database |
| **Jinja2** | Template engine for rendering dynamic HTML pages |
| **Bootstrap 5** | Frontend styling and responsive design |
| CSS | For adding background images and basic styling. |
| **Flask-Login** | User authentication and session management |
| Date-Time | Python Library for managing date&time. |
| **SQLite** | Lightweight local database for storing user data |

---

# 5. Database Schema / ER Diagram

# Models and Relationships:

### a. User

Base model for authentication. Stores name, email, password, and role.

**Relationships:**

- One-to-one with **Doctor**
- One-to-one with **Patient**

## b. Department

Represents medical departments.
**Relationships:**

- One-to-many with **Doctor**

## c. Doctor

Extended profile of User. Stores department and experience.
**Relationships:**

- One-to-one with **User**
- Many-to-one with **Department**
- One-to-many with **Appointment**
- One-to-many with **DoctorAvailability**
- One-to-many with **PatientHistory**

## d. Patient

Extended profile of User.
**Relationships:**

- One-to-one with **User**
- One-to-many with **Appointment**
- One-to-many with **PatientHistory**

## e. Appointment

Connects patients and doctors; stores date, time, and status.
**Relationships:**

- Many-to-one with **Patient**
- Many-to-one with **Doctor**

- One-to-one with **PatientHistory**

## f. PatientHistory

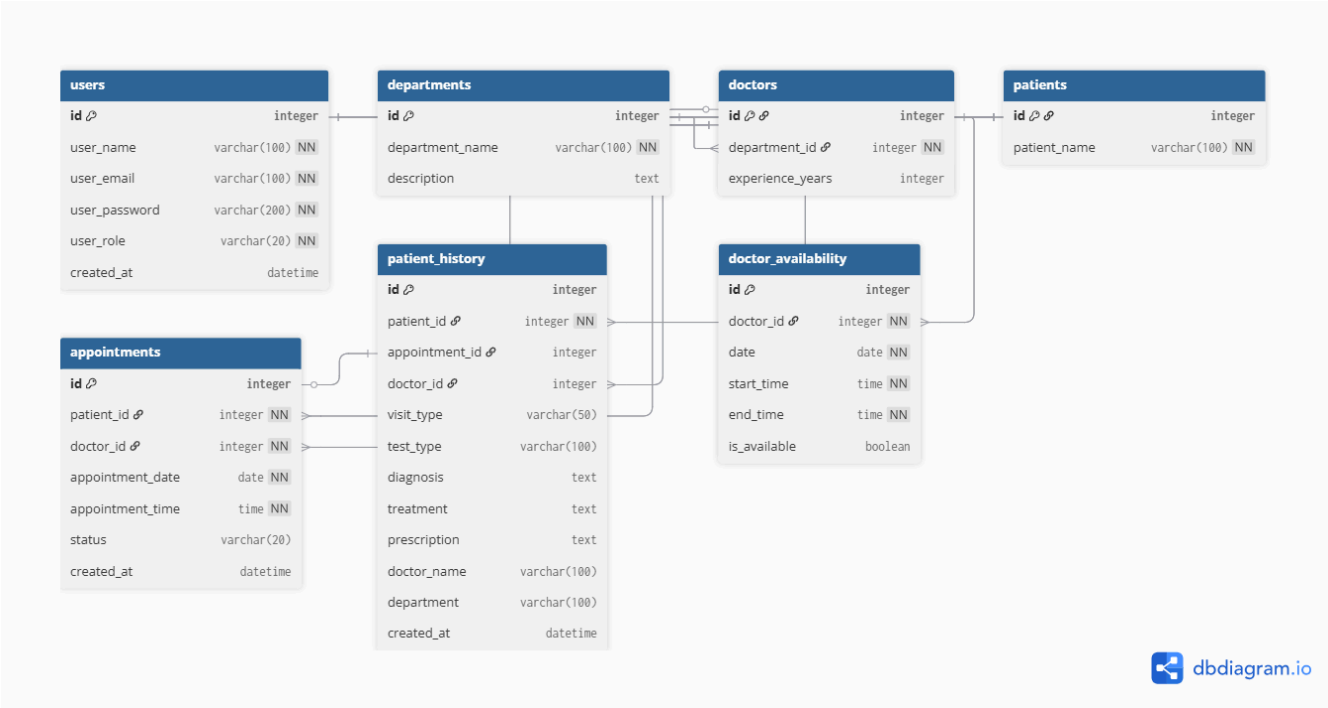
Stores diagnosis, treatment, prescription, and visit details.
**Relationships:**

- Many-to-one with **Patient**
- Many-to-one with **Doctor**
- One-to-one with **Appointment**

## g. DoctorAvailability

Stores available time slots for doctors.
**Relationships:**

- Many-to-one with **Doctor**



# 6. Architecture and Features (optional)

### Architecture Overview:

- **app.py** – main Flask application entry point
- **/models** – database models using SQLAlchemy
- **/routes** – Flask Blueprints for user and activity routes
- **/templates** – Jinja2 HTML templates
- **/static** – CSS, Images used in application

### Implemented Features:

- **Admin dashboard** to manage doctors, departments, and appointments.
- **Doctor dashboard** to view, complete, and manage appointments.
- Doctors can **record diagnosis, treatment, and prescriptions** for patients.
- **Patient dashboard** to book, cancel, or reschedule appointments easily.
- Patients can **view their medical history and prescriptions** anytime.
- **Doctor availability management** for scheduling time slots over 7 days.
- **Appointment booking validation** prevents double-booking and time conflicts.
- **Centralized medical history system** linking patients, doctors, and appointments.
- **Flash messages** for real-time success, error, and warning notifications.
- **Responsive Bootstrap 5 interface** with consistent layout and design.

---

# 7. Video Presentation

**Drive Link:**
24f3004415-HMS project.mp4

---