App Dev – 1
Project Report

1. Student Details

Name: M A Sheik Mansoor
Roll Number: 24F2008027
Email: 24f2008027@ds.study.iitm.ac.in

About Me:
I am a Student at the IIT Madras BS Degree program with a deep interest in web application development. I enjoy building interactive applications that combine learning, analytics, and user experience. My focus is on designing systems that are simple and user-friendly

2. Project Details

Project Title: Hospital Management Systems

Problem Statement:

To design and build a web-based **Hospital Management System (HMS)** application that allows **Admins, Doctors, and Patients** to interact with the system based on their roles. Patients register themselves and then check availability, book appointments, view patient history. Doctors provide availability, view patients past records and provide diagnosis and prescription for treatments. Admin manages doctors, patients, appointments, create departments and register doctors in it.

Approach:

The application was developed using **Flask** with a modular backend structure and a clean separation between routes, models, and templates.
Patients can:

- Sign up and Login into the application.
- Edit there profile and view their past history.
- Search doctors and departments.
- Check availability, Book appointment and Cancel appointment.

3. AI/LLM Declaration

I used **ChatGPT (GPT-5)** to:

- Assist in the logic of creating check availability, book appointments and search.
- Little assistance in debugging.

Extent of usage: ~10–15%, limited to code suggestions.
All core logic and final implementation were done manually.

4. Technologies and Frameworks Used

Technology / Library	Purpose
Flask	Core backend web framework
SQLAlchemy	ORM for SQLite database
Jinja2	Rendering dynamic HTML templates
Javascript	For booking and providing appointments
CSS	For styling the templates
SQLite	Lightweight embedded database

5. Database Schema / ER Diagram

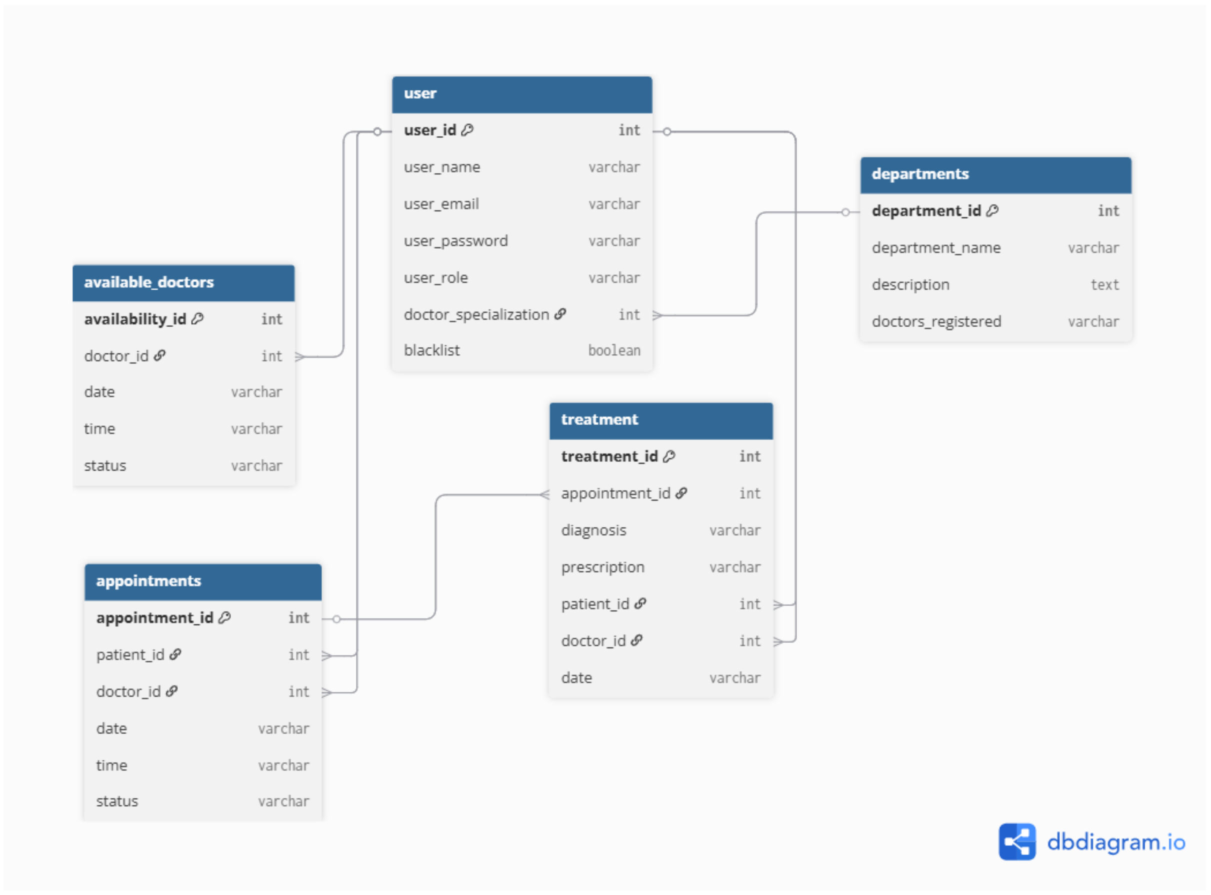
Tables:

- **User** - stores user profile details
  (user_id, user_name, user_email, user_password, user_role, doctor_specialization, blacklist)
- **Department** - stores department details
  (department_id, department_name, description, doctors_registered)
- **Appointments** - stores appointment details
  (appointment_id, patient_id, doctor_id, date, time, status)
- **Treatment** - stores treatment details for appointments
  (treatment_id, appointment_id, diagnosis, prescription, patient_id, doctor_id, date)
- **Availability** - stores doctor availability slots
  (availability_id, doctor_id, date, time, status)

Relationships:

- **One-to-Many → Department → User**
  (One department can have many doctors.)
- **One-to-Many → User (patient) → Appointments**
  (One patient can have many appointments.)

- **One-to-Many → User (doctor) → Appointments**
  (One doctor can have many appointments.)
- **One-to-Many → Appointments → Treatment**
  (One appointment can have many treatments.)
- **One-to-Many → User (doctor) → Availability**
  (One doctor can have multiple availability slots.*)*



6. Architecture and Features

Architecture Overview:

- **app.py** - main Flask application entry point
- **/templates** - Jinja2 HTML templates
- **Instances** – SQLite Database

Implemented Features:

- User Registration & Login
- Search doctors and Departments.
- Provide availability and Book appointments.
- Update and View patient history .
- Edit, Blacklist or Delete Patients and Doctors.

7. Video Presentation

Drive Link:
https://drive.google.com/file/d/1Z6vRGRorpmybeg7ntaDF1xNMwLYt-jzR/view?usp=sharing