App Development Project Report

1. Student Details

Name: Soham Bisoi
Roll Number: 24F1001665

Student Email: 24f1001665@ds.study.iitm.ac.in
About Me: I am a student of the IIT Madras BS Degree program, also an extra departmental   employee of India Post , currently pursuing my Diploma level. I have taken the Modern   Application Development 1 (MAD 1) course project, through which I am exploring the f undamentals of web application development and building practical, like this project of Hospital   Management System.

---

2. Project Details

Project Title: Hospital Management System

Problem Statement: To design and build a Hospital Management System (HMS) web application that allows Admins, Doctors, and Patients to interact with the system based on their roles.

Approach:

The app was built using Flask as the backend framework with a modular structure

---

AI/LLM Declaration

I used **Copilot** to assist in writing models.py, creating API documentation samples, and improving variable naming consistency.
The extent of AI/LLM usage is around **22-25%**, limited to **code suggestions and documentation formatting** and also in some others like in html files at around 20-25%.

4. Technologies and Frameworks Used

Technology / Library	Purpose
Flask:	Web framework

SQLAlchemy:	Database ORM
Werkzeug:	Password hashing
Flask-WTF:	Form handling and validation

5. Database Schema / ER Diagram

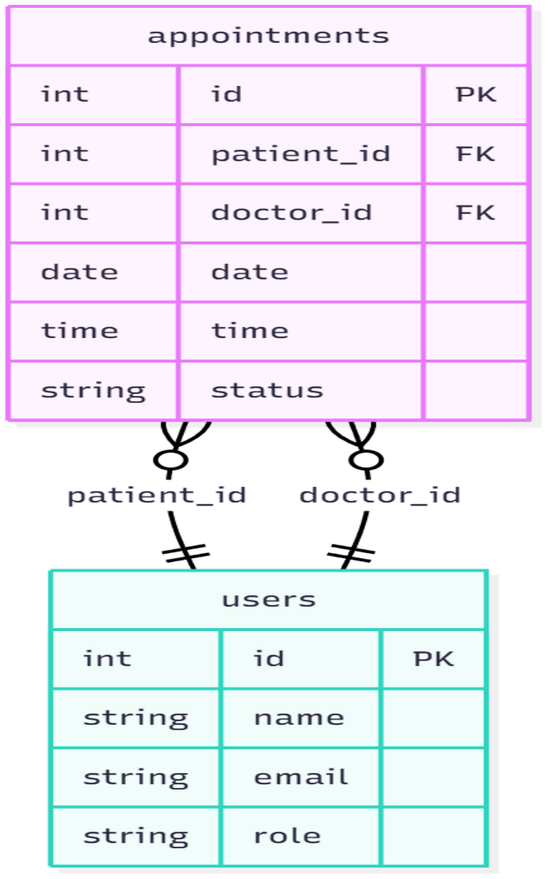

Table: doctors

- **id** → *INTEGER, Primary Key, Auto Increment* — Unique doctor ID.
- **user_id** → *INTEGER, Foreign Key (users.id)* — Links each doctor to a user account.
- **specialization** → *VARCHAR(100), NOT NULL* — Doctor's field of expertise.
- **availability** → *TEXT, NULLABLE* — Available days and time slots.

Table: patients

- **id** → *INTEGER, Primary Key, Auto Increment* — Unique patient ID.
- **user_id** → *INTEGER, Foreign Key (users.id)* — Links each patient to a user account.

- **age** → *INTEGER, NULLABLE* — Patient’s age.
- **gender** → *VARCHAR(10), NULLABLE* — Gender of the patient.
- **blood_group** → *VARCHAR(5), NULLABLE* — Patient’s blood group.
- **address** → *TEXT, NULLABLE* — Address or contact information.

Table: appointments

- **id** → *INTEGER, Primary Key, Auto Increment* — Unique appointment ID.
- **doctor_id** → *INTEGER, Foreign Key (doctors.id)* — Associated doctor.
- **patient_id** → *INTEGER, Foreign Key (patients.id)* — Associated patient.
- **date** → *DATE, NOT NULL* — Appointment date.
- **time** → *TIME, NOT NULL* — Appointment time.
- **status** → *ENUM('Pending', 'Approved', 'Cancelled', 'Completed'), DEFAULT 'Pending'* — Current status of the appointment.

Design Rationale

- **Normalization:** Each entity (User, Doctor, Patient, Appointment) has its own table to avoid redundancy.
- **Reusability:** The users table serves as a base for all user types, simplifying authentication and session management.
- **Scalability:** This design allows easy addition of new roles (like “Nurse” or “Pharmacist”) without structural changes.
- **Data Integrity:** Foreign keys ensure relationships between users, doctors, and patients remain consistent.

---

6. API Resource Endpoints

API Endpoint	Method	Description
`/api/login`	`POST`	Authenticates user credentials
`/api/register`	`POST`	Registers a new patient
`/api/doctors`	`GET`	Retrieves list of all doctors
`/api/appointments`	`GET`	Retrieves appointments for logged-in user
`/api/appointments`	`POST`	Creates a new appointment
`/api/appointments/<id>`	`PUT`	Updates appointment status
`/api/appointments/<id>`	`DELETE`	Cancels an appointment

---

7. Architecture and Features (optional)

Project Organization:-

The project follows the **MVC (Model–View–Controller)** pattern for modularity and clarity.

Folder/File	Description

app.py	Entry point of the Flask application
models/	Contains ORM models for Users, Doctors, Patients, Appointments
routes/	Holds controller files (admin, doctor, patient routes)
templates/	HTML files for different pages using Jinja2 templating
static/	CSS, JS, and image assets
database/	SQLite or MySQL database initialization script
api/	REST API endpoints and logic

Each controller handles user-specific operations (like admin dashboard, doctor appointments, patient bookings.
Templates inherit from a common base.html to maintain a consistent UI across pages.

Implemented Features
Default Features

- User authentication (login/register/logout)
- Role-based access control (Admin, Doctor, Patient)
- Appointment scheduling and management
- Bootstrap 5 responsive UI
- Flash messages for user feedback

Additional / Advanced Features

- Doctor availability tracking
- Profile management for doctors and patients
- Status-based appointment updates (Pending/Approved/Cancelled)
- Admin dashboard for managing all data centrally
- API endpoints for data access and integration

---

8. Video Presentation

Drive Link:
Please click here

Also the link is here -
https://drive.google.com/file/d/1b1ODsoOXNTCCzG3dv6zCsz4MJYL8EY8B/view?usp=sharing

(Accessible to all with "View" permission.)
