App Development Project Report

Modern Application Development 1 Project Report

1. Student Details

Name:	Adithya Sabarish Saravanan
Roll Number:	22f3000018
Email:	22f3000018@ds.study.iitm.ac.in
About Me:	I am online degree student pursuing the BS in data science degree from Germany. I have completed my master's degree with AI specialization in Germany and due to my interest in Mathematics, I am pursuing my career and working now as Cryptographic Algorithm developer at Infineon Technologies.

---

2. Project Details

Project Title: Hospital Management System

Problem Statement:

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

Approach:

The app was built using Flask, SQLALCHEMY and other Jinja libraries as the backend framework with a modular structure. It allows patients and doctors to effectively interact and render functionalities to the user based on their needs. The final visual features are handled by HTML template rendering.

---

3. AI/LLM Declaration

I used **GitHub Co-pilot**:

1. Only to improve the python classes (database relations). Like for example, while defining the backref relationships between relations.
2. HTML templates: To improve the manifestations of Buttons, and for bootstrap based table formatting.
3. The extent of AI/LLM usage is around **10–12%**, limited to **code suggestions and style formatting**.
   All final implementation logic, debugging, and integration were done manually.

---

4. Technologies and Frameworks Used

Technology / Library	Purpose
Flask	Core backend web framework
SQL Alchemy	Object Relational Mapper for SQLite database
Jinja2	Template engine for rendering dynamic HTML pages
Bootstrap 5	Frontend styling and responsive design
SQLite	Lightweight local database for storing user data

---

5. Database Schema / ER Diagram

Tables:

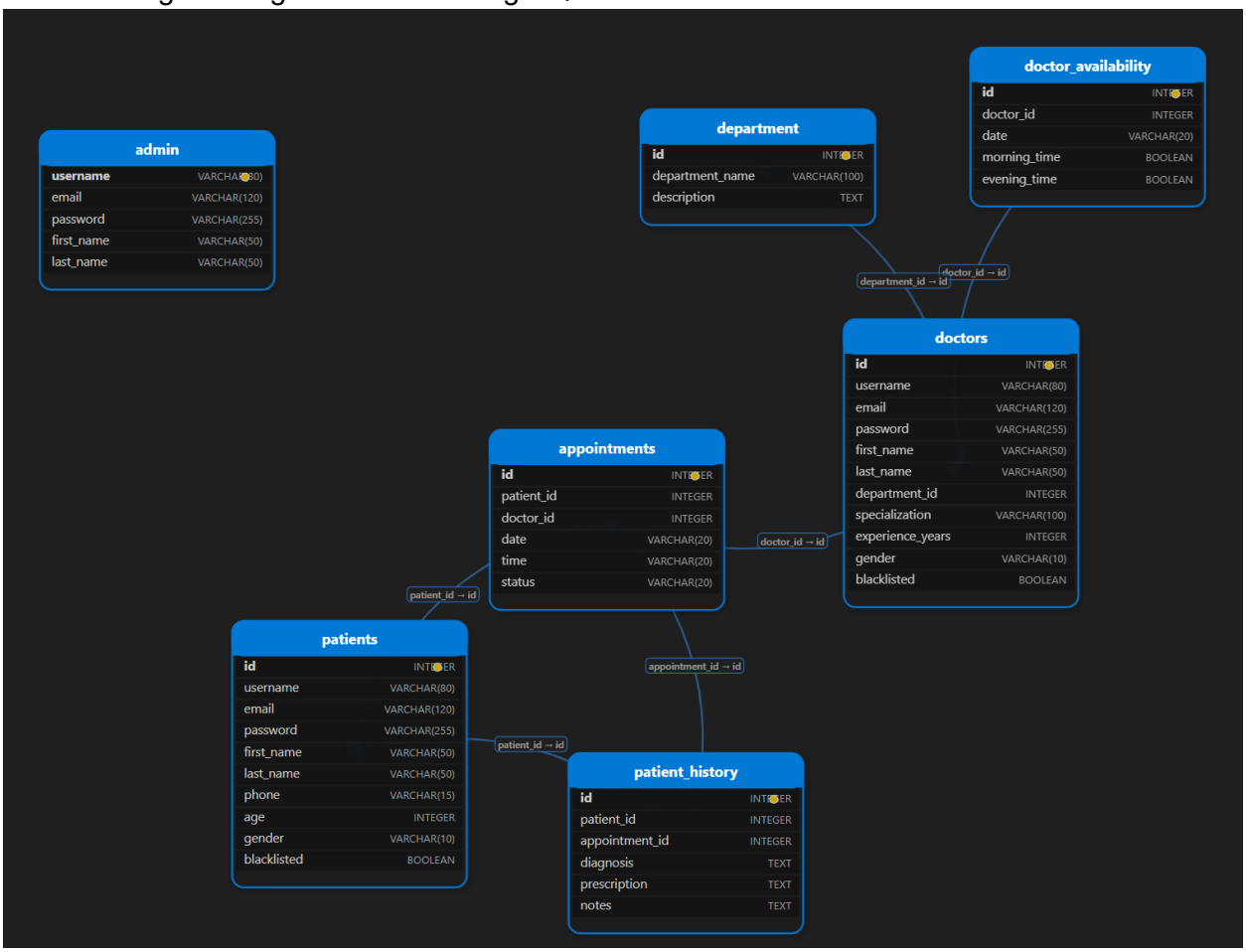
1. **Admin** — stores admin details (id, name, email, password)
2. **Doctor**— store doctors' details (id, username, mail, password, name, blacklisted, other additional details)
3. **Patient**— stores patient information (id, username, mail, password, name, blacklisted, other additional details)
4. **Department**— stores different department information (id, name, description)

5. **Appointment**— contains appointment details between doctor and patient. (doctor id, patient id, date, time, status)

6. **Doctor availability** — To save the availability that the doctor provides to enable appointment bookings by patient. (id, doctor id, date, morning time and evening time)

7. **Patient history** — To store patient treatment history provided by doctor. (id, patient id, appointment id, diagnosis, prescription, notes)

The following ER diagram created using **SQLite Intelli View** VS code extension**:**



Relationships:

- Many-to-one: Doctor → Department: Many doctors belong to one department.
- Many-to-one: Appointment → Patient: Many appointments for one patient.
- Many-to-one: Appointment → Doctor: Many appointments for one doctor.
- One-to-many: Department → Doctor: One department has many doctors.
- Many-to-one: Doctor availability → Doctor: Many availabilities for one doctor.
- Many-to-one: Patient history → Patient: Many histories for one patient.

- One-to-one: Patient history → Appointment and Appointment → Patient history: Each patient history is linked to one appointment, and each appointment can have at most one patient history.

---

6. Architecture and Features (optional)

Architecture Overview:

- **app.py** – main Flask application entry point
- **/application** – database models in models.py, functionalities of admin, doctor and patient and then login/registration handling in a modularized manner.
- **/instance** – contains sqlite3 database
- **/templates** – Jinja2 HTML templates
- **/static** – CSS styling files

Implemented Features:

1. **Admin:**
    a. Add, update, delete, and remove/blacklist doctor/patient profiles (name, specialization, availability).
    b. View and manage all appointments.
    c. Search for patients or doctors by name/specialization.
2. **Doctor:**
    a. Mark appointments as *Completed* or *Cancelled*.
    b. Provide their availability for the next 7 days.
    c. Update patient treatment history like provide diagnosis, treatment and prescriptions.
3. **Patient:**
    a. View their own treatment history and appointment history with diagnosis and prescriptions.
    b. Book as well as cancel appointments with doctors.
    c. Register and login themselves.

Additional Features:

- **Flash message** that the user can voluntarily close after performing an operation. Example – After successfully performing registration, appointment booking, appointment canceling, etc.

---

7. Video Presentation

Drive Link: URL
