---

**Project Report: Hospital Management System**

---

**Author**


- **Name:** Jaii Mukundh A
- **Roll Number:** 24f2000986
- **Email:** 24f2000986@ds.study.iitm.ac.in

  I am passionate about the capabilities of Artificial Intelligence and mainly am focused on how it can be leveraged to bring about a positive change in society.


---

**Description**

This project is a role based hospital management system which consist of Admin functionalities to add doctors and overlook operations. Doctor functionalities to set timeslots and dates for availability. Patient functionalities include scheduling an appointment by choosing a department, doctor and timeslot.

---

**Technologies Used**

- **Flask:** A python web framework that is lightweight.
- **Flask-SQLAlchemy:** Used as an Object-Relational Mapper (ORM) with python.
- **Flask-Login:** Provided a robust and secure solution for managing user sessions.
- **Flask-WTF:** Integrated to create secure web forms, providing essential features like CSRF (Cross-Site Request Forgery) protection and input validation.
- **SQLite:** File based database engine used for production validation.
- **Bootstrap 5:** A popular CSS framework to build apps quickly.
- **Chart.js:** To provide clear and effective visualization in the admin functionalities.

---

**DB Schema Design**

The database schema is designed using a normalized, relational structure to ensure data integrity and prevent redundancy.

- **Admin**: id (Integer, Primary Key), username (String, Unique), password_hash (String)
- **Doctor**: id (Integer, PK), name (String), dept_id (Integer, FK to Department), email (String, Unique), contact_num (String), password_hash (String)
- **Patient**: id (Integer, PK), name (String), email (String, Unique), contact_num (String), password_hash (String)
- **Department**: id (Integer, PK), dept_name (String, Unique)

- **Appointment**: id (Integer, PK), patient_id (Integer, FK to Patient), doctor_id (Integer, FK to Doctor), appt_datetime (DateTime), status (String)
- **Treatment**: id (Integer, PK), appointment_id (Integer, FK to Appointment, Unique), diagnosis (Text), prescription (Text)
- **DoctorAvailability**: id (Integer, PK), doctor_id (Integer, FK to Doctor), day_of_week (String), start_time (Time), end_time (Time)

**Design Rationale:**
The schema is normalized to separate distinct entities. For instance, Treatment is a separate table linked one-to-one with Appointment to keep the appointment model clean. The DoctorAvailability table was created to provide a flexible way for doctors to manage complex weekly schedules without cluttering the main Doctor table. Foreign key constraints (FK) are used extensively to maintain relational integrity—for example, an appointment cannot exist without a valid patient and doctor.

---

**API Design**

These APIs were implemented as standard Flask routes that return JSON data:

- **/api/doctors_by_department/<dept_id>**: Triggered when a patient selects a department in the booking form.
- **/api/available_slots/<doctor_id>/<date_str>**: Called when a patient selects a doctor and a date.
- **/api/chart_data/admin**: Supplies the Admin dashboard with system statistics (total doctors, patients, appointments) in a JSON format that is consumed by Chart.js to render the overview chart.

---

**Architecture and Features**

The project follows a logical and organized structure to separate concerns:

- **app.py**: The main controller of the application. It contains all the route definitions, business logic, and ties together the models, forms, and templates.
- **models.py**: Defines the entire database schema using SQLAlchemy model classes. This is the single source of truth for our data structure.
- **forms.py**: Contains all form definitions using Flask-WTF, centralizing form logic and validation rules.
- **templates**: This directory holds all the HTML files, which are organized into sub-folders by user role (admin, doctor, patient) for clarity and easy maintenance.
- **static**: Contains all static assets, including CSS for styling and JavaScript for front-end interactivity.
- **instance**: This folder is automatically generated to hold the hospital.db SQLite database file, keeping it separate from the source code.

**Key Features Implemented:**

- **Role-Based Access Control:** A robust system that uses decorators (@admin_required, etc.) to protect routes and ensure users can only access pages appropriate for their role.
- **Full CRUD Functionality:** The Admin has full Create, Read, and Delete capabilities for Doctor and Patient management.
- **Dynamic Appointment Booking:** A key feature where the patient-facing form uses JavaScript and internal APIs to provide real-time doctor and time-slot availability, preventing any booking conflicts.
- **Comprehensive Medical Records:** Complete history is maintained for both doctors (to view patient history) and patients (to view their own history).

---

**AI Usage:**

For this project, I was the primary developer, using an AI assistant for specific support tasks. The AI was leveraged to generate initial boilerplate code for the Flask setup and to help debug challenging errors. This assistance streamlined my workflow and is estimated to account for approximately 25-30% of the development effort, while the core application logic, database schema design, and feature implementation were my own work.

---

**Video**

https://drive.google.com/file/d/1Ra4J3MKTe-2wcuUWzUpd8q-JFXg3J38b/view?usp=drive_link