MAD-1 Project Report - MedicAll

1. Student Details

Name: Rohan G
Roll Number: 22f3003138
Email: 22f3003138@ds.study.iitm.ac.in
About Me: I am a dual degree student pursuing my final year in B.Tech Computer Science from IIITDM Kancheepuram and also pursuing B.S Degree in Data Science and Application from IIT Madras. I love technology, AI, backend, system design and my hobbies are to play football.

---

2. Project Details

Project Title: MedicAll - Hospital Management System

Problem Statement:

Create a web-based **Hospital Management System** that helps admins, doctors, and patients manage appointments, doctor details, patient records, and treatment history in one place, using Flask, Jinja2, Bootstrap, HTML and SQLite.

Approach:

I approached the project using **first-principles thinking** - breaking the system down into its core needs: users, appointments, and treatment records. From there, I built each feature step-by-step, starting with database design, then role-based access, and finally connecting the front-end pages with the backend logic to create a smooth workflow for admins, doctors, and patients.

---

3. AI/LLM Declaration

I used **GitHub Copilot** to assist with some code suggestions and **ChatGPT-5** mainly for debugging, making the README.md documentation and this project report.

Category	MAD-1 Allowed %	My Usage %
Backend (Flask + DB)	35%	**20%**
Frontend (HTML/JS)	28%	**26%**
Styling/UI	8%	**6%**
Auth + CRUD Logic	20%	**15%**
Extras (Charts/API/etc.)	9%	**8%**

---

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
Chart.js	To visualize trends such as activity vs. time
Flask-Login	User authentication and session management
SQLite	Relational database management system

Werkzeug	Security helpers (password hashing)
Bootstrap Icons	Icon library for user interface elements

---

5. Database Schema / ER Diagram\

Tables:

- **User — stores user authentication and basic info**
  (id, name, email, password_hash, role, is_active, created_at, updated_at)

- **Department — stores medical departments**
  (id, name, description, created_at)

- **DoctorProfile — stores doctor-specific details**
  (id, user_id, department_id, phone, qualification, bio, is_blacklisted)

- **PatientProfile — stores patient-specific details**
  (id, user_id, phone, dob, gender, address, is_blacklisted)

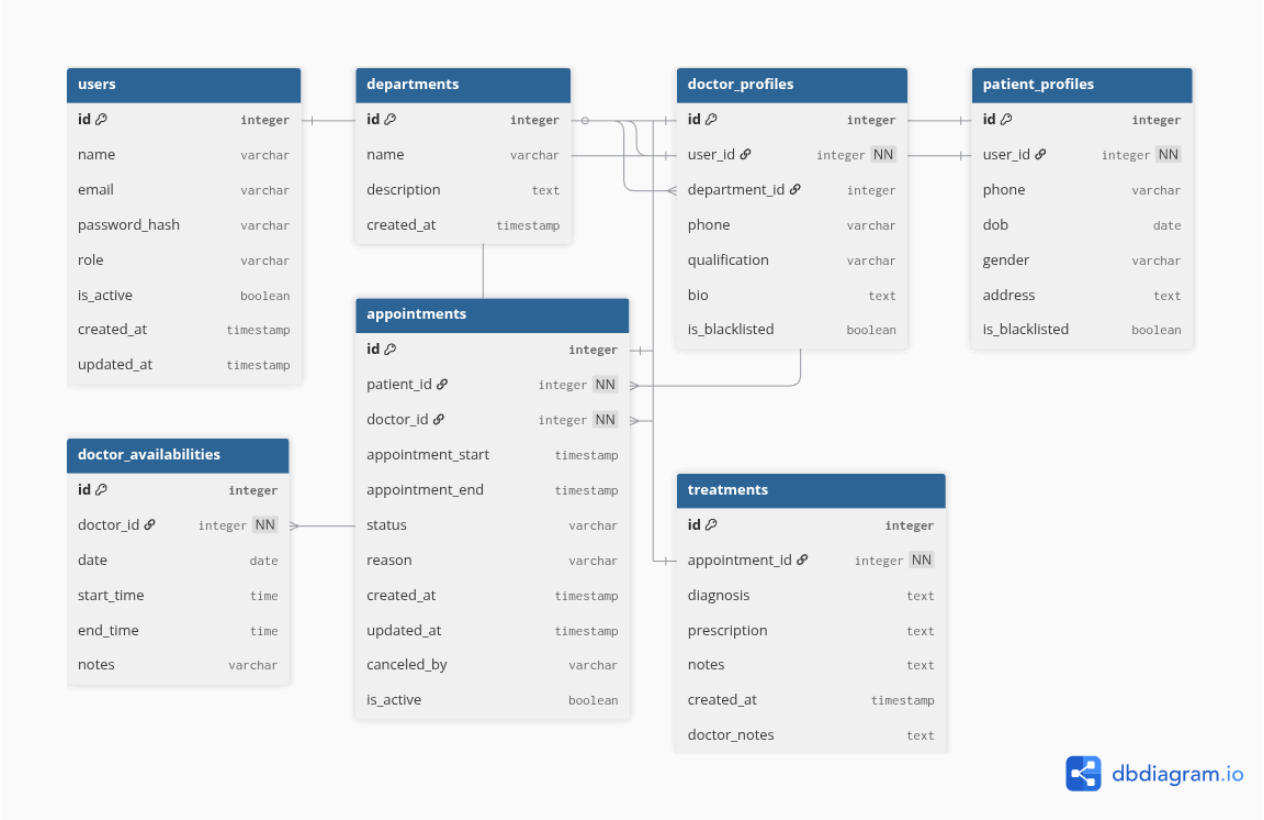
- **DoctorAvailability — stores doctor's available slots**
  (id, doctor_id, date, start_time, end_time, notes)

- **Appointment — stores appointment records**
  (id, patient_id, doctor_id, appointment_start, appointment_end, status, reason, created_at, updated_at, canceled_by, is_active)

- **Treatment — stores medical treatment records for appointments**
  (id, appointment_id, diagnosis, prescription, notes, created_at, doctor_notes)

Relationships:

- One-to-One → User → DoctorProfile
- One-to-One → User → PatientProfile
- One-to-Many → Department → DoctorProfile
- One-to-Many → DoctorProfile → DoctorAvailability
- One-to-Many → DoctorProfile → Appointment
- One-to-Many → PatientProfile → Appointment
- One-to-One → Appointment → Treatment



---

6. API Resource Endpoints

Here are some of the API endpoints -

Endpoint	Method	Description
/api/doctors	GET	Retrieve list of all doctors
/api/doctors/{id}	GET	Fetch details of a specific doctor by ID
/api/patients/{id}	GET	Fetch details of a specific patient by ID
/api/appointments	GET	Retrieve list of all appointments
/api/appointments	POST	Create a new appointment

YAML API Definition File:

Included separately in the submission ZIP as `api.yaml`.

---

7. Architecture and Features

Architecture Overview:

app.py – Main Flask application entry point and configuration
/models – Database models using SQLAlchemy (User, Appointment, Profiles)
/routes – Flask Blueprints for role-based logic (auth, admin, doctor, patient, api)
/templates – Jinja2 HTML templates for dynamic page rendering
/static – CSS, JS, images, and Chart.js visualization files
/utils – Helper functions for input validation and sanitization

Implemented Features:

- Role-based access control (Admin, Doctor, Patient)
- User registration, secure login, and session management
- Doctor availability scheduling and slot management
- Patient appointment booking and history tracking
  Medical treatment recording (Diagnosis, Prescription, Notes)
- Interactive dashboards with Chart.js analytics for all roles
- Responsive UI design using Bootstrap 5
- REST API endpoints for external integration

Additional Features:

- Custom input validation and sanitization
- Search functionality for finding doctors
- Secure password hashing with Werkzeug

---

8. Project Links

Drive Link:
https://drive.google.com/file/d/1RDeyK9acTkWJAViE9fVH6n2j874cvhzK/view?usp=sharing