App Dev Project Report

Hospital Management System (MediCare)
MAD1 – IIT Madras BS Degree

1. Student Details

Name: *Ridhi Sehgal*
Roll Number: *DS24F2008371*
Email: *24f2008371@ds.study.iitm.ac.in*

About Me:
I am a Diploma-level student at IIT Madras, pursuing BS in Data Science and Applications. I have a strong passion for application development, backend engineering, graphic design and UI/UX design. I love creating real-world solutions that blend clean interfaces with functional, reliable backend systems.

---

2. Project Details

Project title : MediCare : Hospital management systems

Problem statement :

You are required to 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 with a modular backend and a clear role-based design. Admins manage hospital data, doctors handle appointments and treatments, and patients book slots and view history. SQLAlchemy models support authentication, scheduling, and record management, while Bootstrap and Jinja provide a simple, user-friendly interface. The overall approach focused on creating a stable backend and an intuitive UI to streamline hospital workflows.

3. AI/LLM Declaration

I, Ridhi Sehgal, declare that **this project was developed primarily through my own learning, coding, and implementation efforts**.
AI tools such as **ChatGPT** were used **only for the following purposes**:

- Debugging errors and understanding error messages
- Getting explanations of Python, Flask, SQLAlchemy, and Jinja concepts and understanding some major backend routes
- Improving documentation clarity and writing quality
- Refining UI/UX ideas and structuring report content and organizing the codes
- Generating small helper snippets or troubleshooting specific issues

Based on my usage, I estimate that **AI/LLMs contributed approximately 15–20%** of the total work, limited strictly to explanation, debugging help, and documentation refinement.

The remaining **80%** of the project — including planning, architecture, backend logic, database modelling, route design, and implementation — was completed independently by me.

AI tools **were NOT used to generate the full project**, database schema, or core logic.
All significant coding, architectural decisions, database design, route handling, and feature implementation were done independently by me.

I understand the IIT Madras academic guidelines regarding the responsible use of AI, and I confirm that **the project reflects my own work**, with AI used only as a supporting tool for learning and enhancement.

4. Technologies and Frameworks Used

Technology / Library	Purpose
Flask	Core backend web framework for routing, logic, and server-side processing
SQLAlchemy	ORM used to manage database tables, relationships, and queries
Jinja2	Template engine for rendering dynamic HTML pages
Bootstrap 5	Frontend styling, layout, and responsive UI design
HTML5 + CSS3	Structure and styling of user-facing web pages
Font Awesome	Icons used for buttons, actions, and UI enhancements
SQLite	Lightweight relational database for storing users, appointments, treatments, etc.
Flask Sessions	Authentication, role-based access, and session management
JavaScript	Interactive frontend behavior and UI responsiveness

5. Database Schema / ER Diagram

Tables:

1. **User** — stores login credentials & role information
   (id, username, password, type, blocked)

2. **Patient** — stores patient profile details
   (id, user_id, name, age, gender, phone, email, blocked)

3. **Doctor** — stores doctor details
   (id, user_id, doctor_name, email, experience, department_id)

4. **Department** — stores hospital departments
   (id, name, description)

5. **Appointment** — stores appointment bookings
   (id, patient_id, doctor_id, date, time, status, visit_type)

6. **Treatment** — stores treatment/visit records
   (id, appointment_id, doctor_id, date, time, diagnosis, prescription, tests_done, medicines, visit_type)

7. **Availability** — stores doctor availability slots
   (id, doctor_id, date, slot, is_available)

8. **Enquiry** — stores messages from the landing page
   (id, name, email, message)

Relationships:

One-to-Many

- User → Patient

- User → Doctor

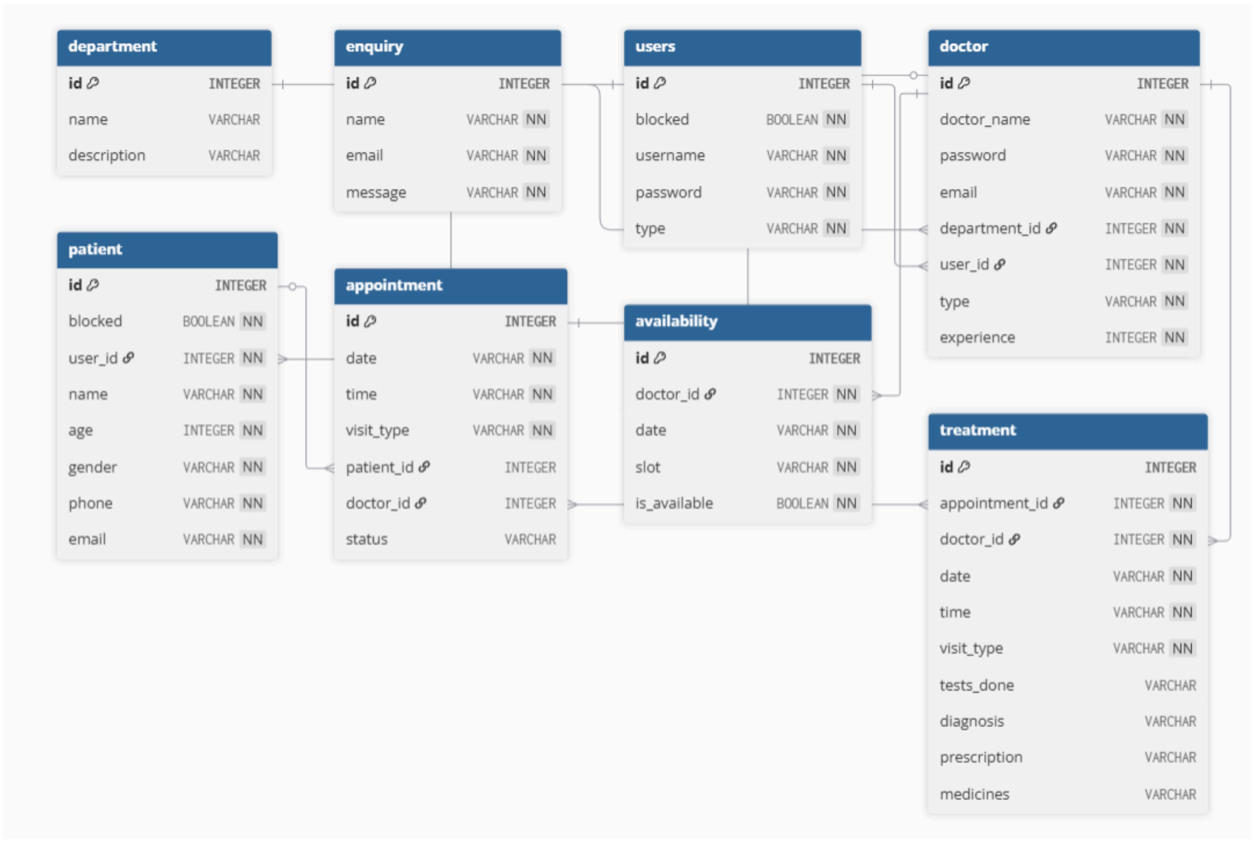
- Department → Doctor
- Patient → Appointment
- Doctor → Appointment
- Doctor → Availability
- Appointment → Treatment

Many-to-One

- Appointment → Patient
- Appointment → Doctor
- Doctor → Department

Cascade Deletes

- Deleting a **User** deletes its linked **Doctor/Patient**
- Deleting a **Doctor** deletes its **Availabilities** and optionally *past* appointments (based on logic)
- Deleting an **Appointment** deletes its **Treatment records**



(db made by dbdiagram.io)

---

6. Architecture and Features

Architecture Overview

- **app.py / controllers.py** – Main Flask application entry point and route handling
- **/models** – SQLAlchemy ORM models for User, Doctor, Patient, Appointment, Treatment, etc.
- **/templates** – Jinja2 HTML templates for all pages (admin, doctor, patient dashboards)
- **/static** – CSS, JS, images, icons, and Bootstrap resources
- **/database** – Database configuration and initialization

- **/application** – Modular structure containing routes, utilities, and backend logic

---

Implemented Features

Authentication & Roles

- User login system with sessions
- Three roles: **Admin**, **Doctor**, **Patient**
- Role-based access and protected routes

Admin Features

- Add, edit, delete doctors
- Add, edit, delete patients
- View all appointments
- Blacklist/unblacklist users
- Manage departments
- View patient history

Doctor Features

- View assigned appointments
- Mark appointments as completed
- Record patient visit details (diagnosis, tests, prescription)
- Manage availability slots (add/delete/clear availability)
- View patient medical history

Patient Features

- Register a new account
- Edit profile
- Browse departments
- View doctor details
- Check doctor availability
- Book appointment slots
- Cancel appointments
- View complete medical history

Appointment Management

- Real-time slot availability
- Status handling (Upcoming, Completed)
- Automatic slot release on cancellation or completion

Landing Page Features

- Enquiry submission form
- Simple public homepage

Additional Features

- Flash messages for all actions (success/error feedback)
- Clean, responsive UI using Bootstrap 5

- Logical database relationships with cascading deletes
- Error handling for login failures and unauthorized access

---

7. API / Route Summary

7. API / Route Summary

Authentication & User Management

Route	Method	Description
`/register`	POST	Register a new patient user
`/login`	POST	Login for admin, doctor, and patient
`/logout`	GET	Logout user and clear session

Admin Panel

Route	Method	Description
`/admin`	GET	Admin dashboard overview
`/add_doctor`	POST	Add a new doctor
`/edit_doctor/<id>`	POST	Edit doctor details
`/delete_doctor/<id>`	GET	Delete doctor safely (no active appointments)
`/blacklist_doctor/<id>`	GET	Block a doctor

`/unblacklist_doctor/<id>`	GET	Unblock a doctor
`/edit_patient/<id>`	POST	Edit patient details
`/delete_patient/<id>`	GET	Delete patient
`/blacklist_patient/<id>`	GET	Block a patient
`/unblacklist_patient/<id>`	GET	Unblock a patient

Doctor Dashboard

Route	Method	Description
`/doctor`	GET	Doctor dashboard (appointments + patients)
`/complete_appointment/<id>`	GET	Mark appointment as completed
`/delete_appointment/<id>`	GET	Delete appointment
`/update_appointment/<id>`	POST	Add treatment + update record
`/doctor/availability`	GET/POST	Add daily availability
`/doctor/availability/delete/<id>`	GET	Delete availability slot
`/doctor/set_availability`	GET/POST	Advanced availability with validations
`/doctor/clear_availability`	GET	Remove all availability slots

Patient Dashboard

Route	Method	Description

`/patient`	GET	Patient dashboard with appointments
`/edit_profile`	POST	Edit patient profile
`/cancel_appointment/<id>`	GET	Cancel booked appointment
`/department/<dept_name>`	GET	Show doctors in selected department
`/doctor/details/<id>`	GET	View doctor profile
`/doctor/check_availability/<doctor_id>`	GET	View doctor’s available slots
`/book_slot/<slot_id>`	POST	Book appointment slot
`/patient_history`	GET	View patient medical history

General / Public

Route	Method	Description
`/`	GET	Landing page
`/submit_enquiry`	POST	Submit enquiry form

---

8. Video Presentation

Drive Link:

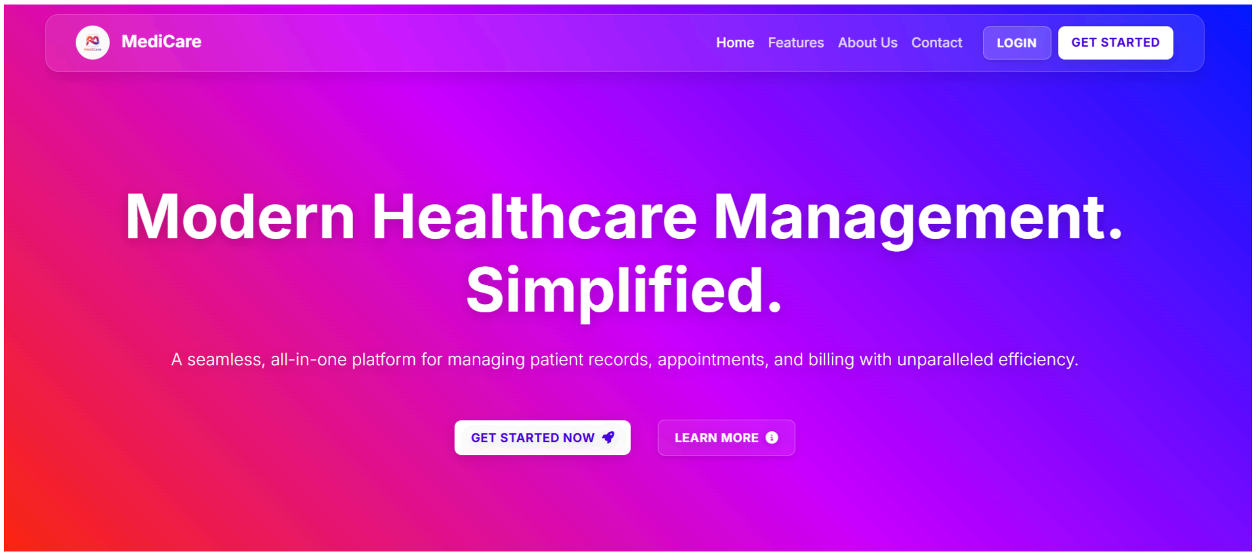
APP DEV PROJECT - RIDHI - VIDEO PRESENTATION

(Accessible to all with “View” permission.)

9. Conclusion

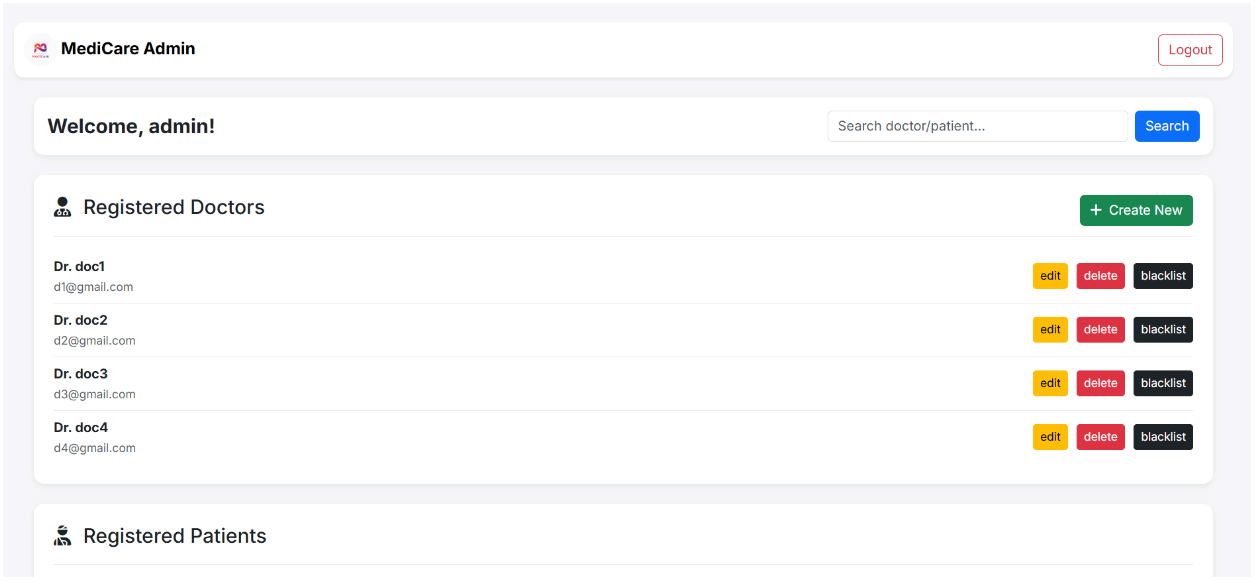
MediCare successfully solves real-world issues in hospital appointment and record management. The system provides a clean, responsive UI and a structured backend that ensures reliability. The project strengthened my understanding of backend systems, database design, and full-stack app development.

HERE ARE SOME UI SCREENSHOTS 😊



Modern Healthcare Management. Simplified.

A seamless, all-in-one platform for managing patient records, appointments, and billing with unparalleled efficiency.

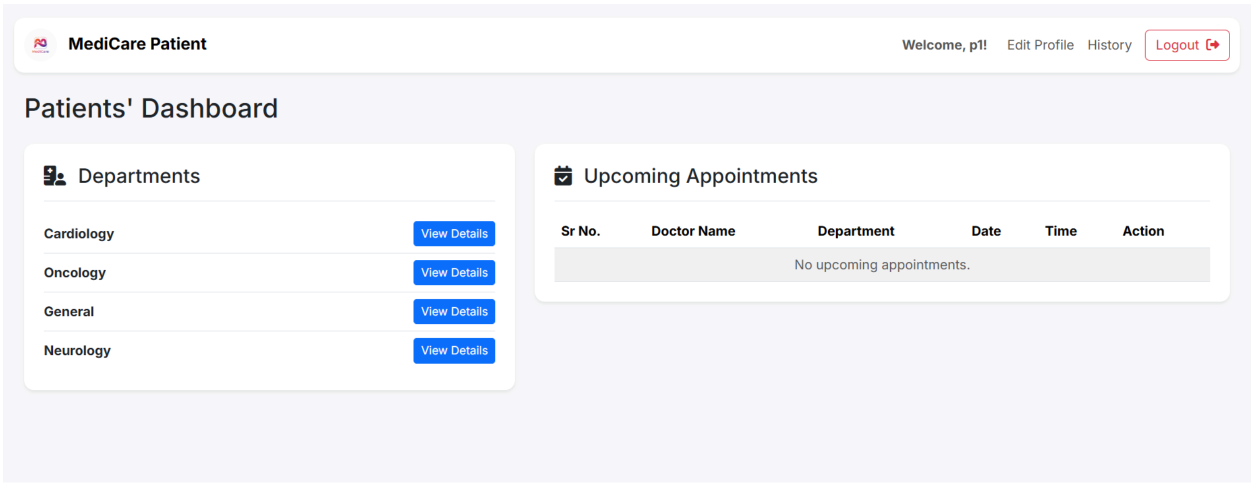


Welcome, admin!

Registered Doctors

Dr. doc1 d1@gmail.com	edit	delete	blacklist
Dr. doc2 d2@gmail.com	edit	delete	blacklist
Dr. doc3 d3@gmail.com	edit	delete	blacklist
Dr. doc4 d4@gmail.com	edit	delete	blacklist

Registered Patients



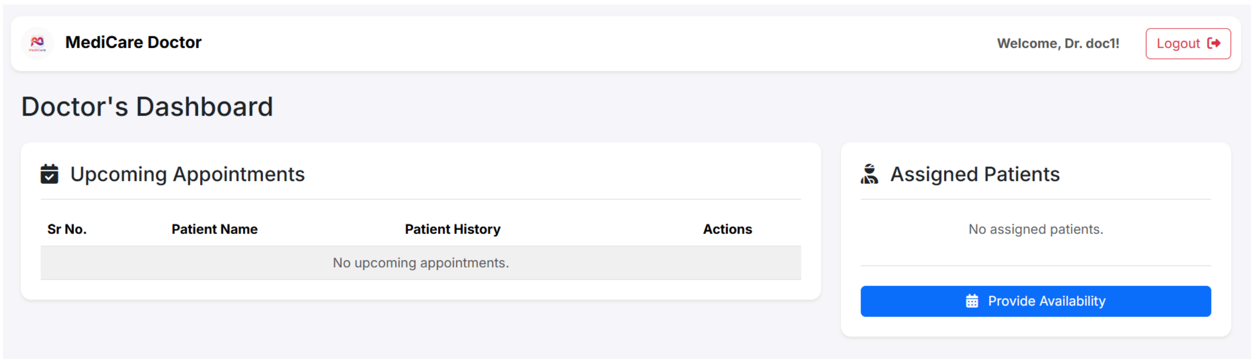
Patients' Dashboard

Departments

Cardiology	View Details
Oncology	View Details
General	View Details
Neurology	View Details

Upcoming Appointments

Sr No.	Doctor Name	Department	Date	Time	Action
No upcoming appointments.					



Doctor's Dashboard

Upcoming Appointments

Sr No.	Patient Name	Patient History	Actions
No upcoming appointments.			

Assigned Patients

No assigned patients.