App Dev Project Report

(NHS - Health_Management_System)

Name: GAYATHRI SALINI

Roll No: 24f3000207

Email: 24F3000207@ds.study.iitm.ac.in

About me: I am a student in the IIT Madras BS Degree program, deeply interested in web app development and data-driven technologies. I enjoy exploring new ideas, creating useful tools, and constantly improving my skills as I build projects that solve problems.

Project Details

Project Title: Hospital Management System

Problem Statement: The Hospital Management System is a web application designed to help hospitals efficiently manage patients, doctors, appointments, and treatments.

Approach: I developed a web application using Flask and SQLAlchemy, implementing role-based access for admin, doctors, and patients. Core features include authentication, appointment scheduling, treatment records, and an analytics dashboard with charts.

AI/LLM Declaration

I used AI for conceptual clarification, including understanding Flask Blueprints, help with reading parts of documentation that were difficult to understand. Additionally, AI provided guidance on some integration points, including how to insert charts (3%) for dashboard visualizations. Beyond these, you did not use AI for other parts like debugging or core implementation.

Charts(3%),serverside-validation(1%),test-debugging(<1%)

Technologies and Frameworks Used

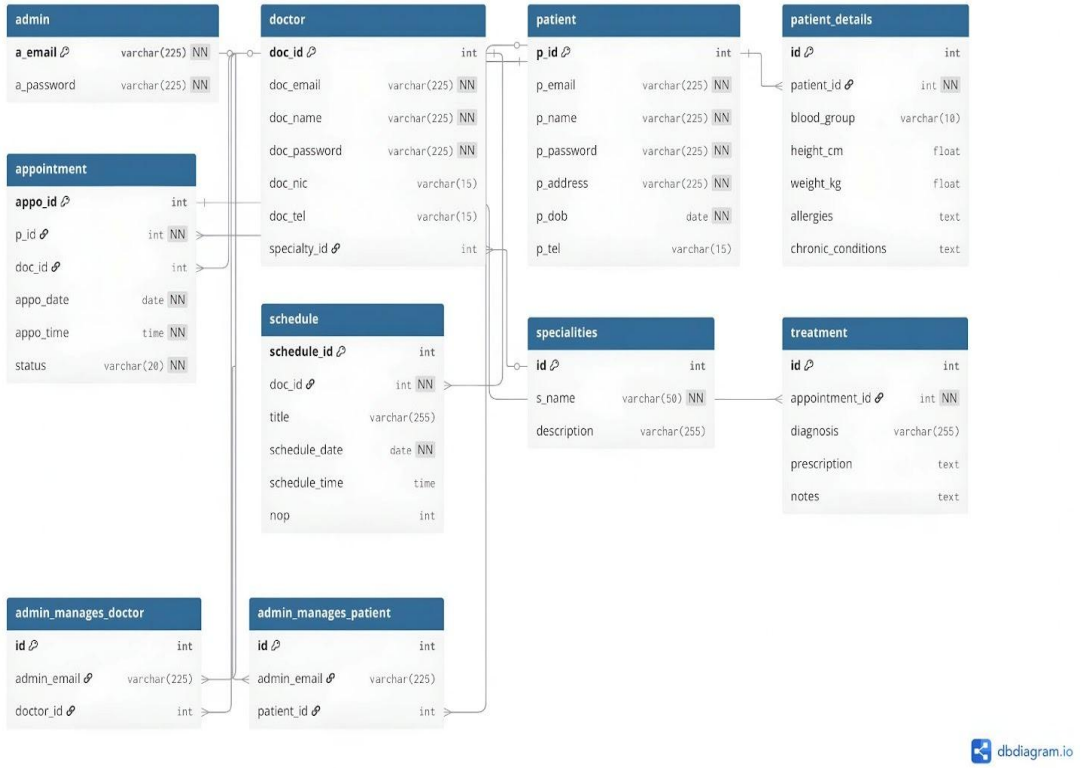
Technology/Library	Purpose
Flask	Core backend web framework
SQLAlchemy	Object Relational Mapper for SQLite database, Manage models and migrations
SQLite	Local data storage
Jinja2	Template engine for rendering dynamic HTML pages
Chart.js	To visualize data visualizations
Flask-Login	User authentication and session management
WTForms	Frontend form validation
Flask Blueprints	Organize application routes for better structure
Flask- Migrate	Manages database schema migrations seamlessly with SQL Alchemy models

Database Schema / ER Diagram

Table Names	Attributes	Description & Relationships
Admin	a_email, a_password	Stores admin login info
Doctor	doc_id, doc_email, doc_name, doc_password, doc_nic, doc_tel, specialty_id	Stores doctor details; specialty_id → specialities (Many-to-One)
Patient	p_id, p_email, p_name, p_password, p_address, p_dob, p_tel	Stores patient profile info
Patient_details	id, patient_id, blood_group, height_cm, weight_kg, allergies, chronic_conditions	patient_id → patient (One-to-One)
Appointment	appo_id, p_id, doc_id, appo_date, appo_time, status	p_id → patient (Many-to-One), doc_id → doctor (Many-to-One)
Doc-schedule	schedule_id, doc_id, title, schedule_date, schedule_time, nop	doc_id → doctor (Many-to-One)
Specialities	id, s_name, description	Doctor specialty
Treatments	id, appointment_id, diagnosis, prescription, notes	appointment_id → appointment (Many-to-One)

ER Diagram

Project brief: [View here (Database ER Diagram)]



Architecture and Features

- app.py – main Flask application entry point
- /models – database models using SQL Alchemy
- /routes – Flask Blueprints for admin, doctor, patient
- /templates – Jinja2 HTML templates for all user interfaces
- /static – CSS, JavaScript files, chart visualizations, and images
- /migrations – database migration scripts managed by Flask-Migrate
- /forms – WTForms classes for form handling and validation

Implemented Features:

- User registration and login for admin, doctor, and patient roles
- Role-based dashboard views and access control
- Doctor management (add, edit, assign specialities and schedules)
- Patient management (add, edit, view detailed profiles)
- Appointment booking and management system
- Treatment record management for doctors
- Analytics dashboard with data visualizations using Chart.js
- Summary statistics and status breakdown (doctors, patients, appointments,treatments)
- Secure password hashing and user session handling with Flask-Login
- WTForms-based form validation

Video Presentation

Video_Report(Drive Link) : Website Video(report).wmv