App Dev Project Report

1. Student Details

Name: Harsh Raj
Roll Number: 24F2004956
Email: 24f2004956@ds.study.iitm.ac.in
About Me: I am a dual-degree student pursuing B.Tech(4th semester) in Computer Science and Engineering from Bihar, along with the IIT Madras BS Degree program(Diploma level). I have a strong interest in web application development and data-driven technologies. I enjoy creating meaningful applications that blend learning, analytics, and user experience. When I'm not in front of my screen, you'll likely find me playing volleyball.

---

2. Project Details

Project Title: "Hospital Management System"

Problem Statement:

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 Hospital Management System was developed with a role-based access control model for Admins, Doctors, and Patients. I designed wireframes, the database schema, and API routes from the project documentation, then implemented the backend first using Flask—exposing the core APIs and enforcing RBAC on protected endpoints. Backend functionality was validated using Insomnia. After the API surface was stable, I built the frontend in Vue 3, integrating with the Flask APIs and iterating on the UI based on the wireframes. The application was manually tested after each feature implementation to ensure correctness and usability.

---

3. AI/LLM Declaration

I used **ChatGPT (GPT-5)** to assist in writing Vue template and style section for looking modern, creating API documentation samples, and improving variable naming consistency.

The extent of AI/LLM usage is around **20–25%**, limited to **code suggestions and documentation formatting**.
All final implementation logic, debugging, and integration were done manually.

---

4. Technologies and Frameworks Used

Technology / Library	Purpose
Jinja2	Template engine for rendering dynamic HTML pages
Bootstrap 5	Frontend styling and responsive design
Chart.js	To visualize trends such as department trend
SQLite	Lightweight local database for storing user data
Flask	Core backend framework for building RESTful API services
Flask-JWTExtended	Authentication and role-based access control (RBAC) using JWT
Flask-SQLAlchemy	ORM for managing database models and operations
SQLAlchemy	Core database engine underlying Flask-SQLAlchemy
Flask-CORS	Enables secure communication between Vue 3 frontend and Flask backend
Vue 3	Frontend JavaScript framework used for building interactive UI components
Celery	Handles background tasks such as asynchronous operations
Redis	Message broker and caching layer used by Celery
ReportLab	Generates PDF documents such as reports or patient summaries
python-dateutil/pytz	Date and time processing utilities

---

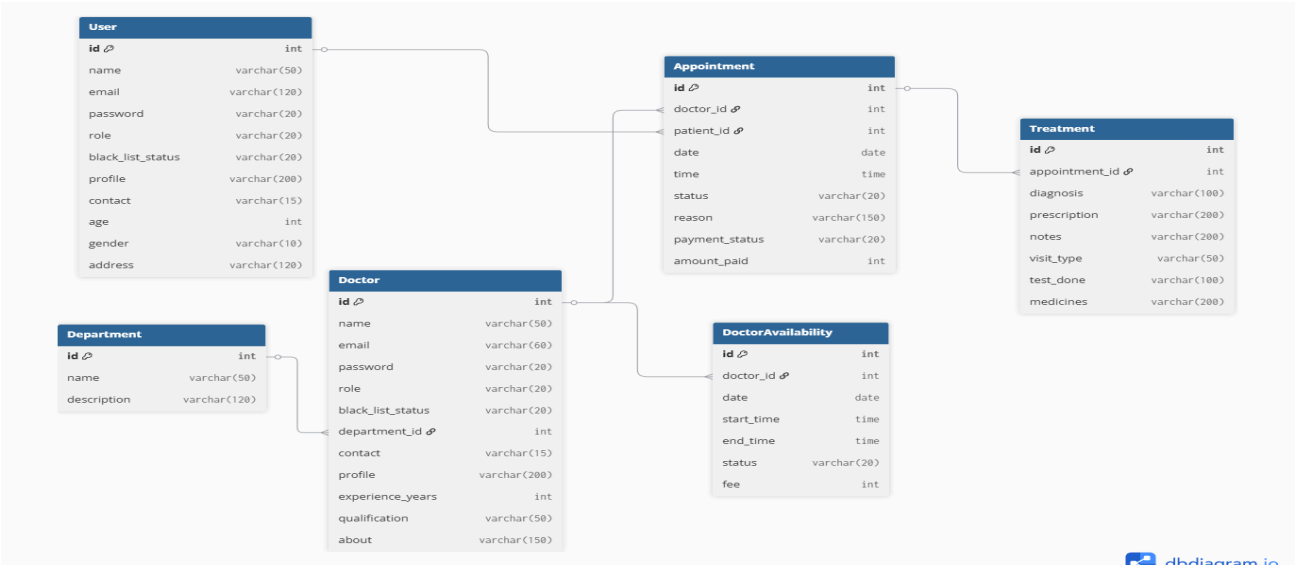
5. Database Schema / ER Diagram

Tables:

1. **User** — stores patient profile details
   (id, name, email, password, role, black_list_status, profile, contact, age, gender, address)
2. **Doctor** — stores doctor details and professional information
   (id, name, email, password, role, black_list_status, department_id, contact, profile, experience_years, qualification, about)
3. **Department** — stores hospital departments
   (id, name, description)
4. **DoctorAvailability** — stores available timings of each doctor
   (id, doctor_id, date, start_time, end_time, status, fee)
5. **Appointment** — stores appointment booking details
   (id, doctor_id, patient_id, date, time, status, reason, payment_status, amount_paid)
6. **Treatment** — stores treatment and medical records for each appointment
   (id, appointment_id, diagnosis, prescription, notes, visit_type, test_done, medicines)

Relationships:

- **One-to-Many → User → Appointment**
  (A user/patient can have many appointments)
- **One-to-Many → Doctor → Appointment**
  (A doctor can have many appointments)
- **One-to-Many → Department → Doctor**
  (A department can have many doctors)
- **One-to-Many → Doctor → DoctorAvailability**
  (A doctor can have many availability slots)
- **One-to-Many → Appointment → Treatment**
  (Each appointment can have multiple treatment records)



6. API Resource Endpoints

Endpoint	Method	Description
`/api/welcome`	GET	Welcome route for basic API check
`/api/login`	POST	Authenticate user and return session/token
`/api/account/<account_type>/<account_id>`	GET	Fetch account details (patient/doctor/admin)
`/api/register`	POST	Register a new user (patient/doctor based on request)
`/api/doctor`	GET / POST	Fetch all doctors / Add new doctor
`/api/doctor/<doctor_id>`	GET / PUT / DELETE	Fetch, update, or delete doctor details
`/api/doctor/me`	GET	Fetch logged-in doctor's profile
`/api/doctor/profile-update`	PUT	Update doctor’s profile information
`/api/doctor-details/<doctor_id>`	GET	Get doctor details with department & availability
`/api/patient`	GET / POST	Fetch all patients / Add new patient
`/api/patient/<patient_id>`	GET / PUT / DELETE	Fetch, update, or delete patient details
`/api/patient/profile-update`	PUT	Update patient profile
`/api/department`	GET / POST	Fetch all departments / Add new department
`/api/department/<department_id>`	GET / PUT / DELETE	Department CRUD operations
`/api/appointment`	GET / POST	Fetch appointments / Book appointment

`/api/appointment/<appointment_id>`	GET / PUT / DELETE	Appointment details, reschedule, or cancel
`/api/appointments/<appointment_id>/pay`	POST	Make appointment payment
`/api/treatment/<appointment_id>`	POST	Add treatment for an appointment
`/api/treatment/doctor`	GET	Doctor's treatment history
`/api/treatment/patient`	GET	Patient's treatment history
`/api/treatment/admin`	GET	Full treatment history for admin
`/api/treatment/export`	GET	Export treatment data (CSV/Excel)
`/api/availability`	GET / POST	Fetch or add doctor availability
`/api/availability-week`	GET	Weekly availability for doctor
`/api/report/appointments-trend`	GET	Appointment trend analytics
`/api/report/department-demand`	GET	Department demand analytics
`/api/reports/monthly`	GET	Generate & download monthly PDF report

YAML API Definition File:
Included separately in the submission ZIP as `api.yaml`.

7. Architecture and Features (optional)

Architecture Overview:

- **/backend/app.py** – main Flask application entry point
- **/backend/application/models.py** – database models using SQLAlchemy
- **/backend/application/task.py** – celery task daily/monthly emails and export
- **/backend/application/api** – All backend API routes
- **/frontend** – in this file run npm run dev to start frontend server
- **/frontend/public** – app icons
- **/frontend/src/components** – all frontend Vue pages
- **/frontend/src/router/index.js** –all routes mounted

Implemented Features:

- Login & registration system for Patients; dedicated login for Doctor and Admin.
- Role-Based Access Control (RBAC) implemented to manage Admin, Doctor, and Patient permissions.
- Secure session/token–based authentication.
  Pre-created Admin account (no admin registration allowed).
  Admin dashboard displaying total doctors, patients, and appointments.
- Add, update, view, and manage doctor profiles.
- View all upcoming and past appointments.
  Search doctors/patients by name, ID, specialization, or contact details.
  Edit patient details if needed.
- Blacklist/remove doctors or patients.
- View department–wise appointment reports & demand analytics.
- Doctor dashboard showing today's and weekly upcoming appointments.
- View ass igned patients list.
- Update appointment status (Booked → Completed → Cancelled).
- Add diagnosis, treatment details, prescriptions, and next-visit recommendations.
- Provide availability schedule for the next 7 days.
- View complete treatment history of their patients.
- Patient self-registration and login.
- Dashboard showing specialties/departments.
- View doctor availability for the coming week.
- Book and cancel appointments.
- View upcoming appointments with status.
- Access full past treatment history (diagnosis, prescription, doctor notes).
- Edit/update personal profile.
- Prevent double booking for doctor on same date/time.
- Automatic appointment status updates.
- Store complete treatment records linked to appointments.
- **Daily Reminder Job:** Sends reminders via email/SMS/chat for same-day appointments.
- **Monthly Doctor Activity Report:** Generates HTML/PDF report of monthly appointments, treatments, and diagnosis and sends via email.

- **User-triggered CSV Export:** Patients can export their complete treatment history through an async background job with success notification.
- Caching added for repeated queries and report data.
- Cache expiry implemented for optimal performance.
- Optimized API response times.
- Fully tested API endpoints using Insomnia.
- Complete CRUD workflows for admin, doctor, patient, department, appointment, and treatment modules.
- Clean and responsive UI built using Vue 3.
- Form validation and error handling on both frontend and backend.

Additional Features:

- PDF generation for monthly reports.
- CSV export for doctor/patient treatment histories.
- Appointment payment simulation (dummy payment portal).
- Charts and analytics.
- Fully responsive layout suitable for mobile and desktop.
- Department demand analytics and appointment trends reporting.
- Search functionality across multiple entities (doctor, patient, department).
- User-friendly wireframes and structured database schema implemented.

---

8. Video Presentation

Drive Link:
https://drive.google.com/file/d/1EWCo9RjpczWqsO0iBksqUh9lTSyo_Oi9/view?usp=drivesdk

---