Project Report

Author

Harshit Sahu

23f2001635

23f2001635@ds.study.iitm.ac.in

[A couple of lines about you]

AI/LLM Usage Declaration

I used **Gemini 3 Pro** to assist with code generation, debugging, and refactoring.

- **Extent of Use:** Approximately 15-20%.
- **Primary Use Cases:** Boilerplate code, and troubleshooting specific errors.
- **Extra Details:** Used for generating initial project structure, writing Celery task logic, and debugging API issues.

Description

This project involves building a full-stack **Hospital Management System** to manage doctors, patients, appointments, and departments. The core requirements included implementing role-based access control, setting up background jobs for notifications/reports, and creating interactive dashboards for different user types.Technologies Used

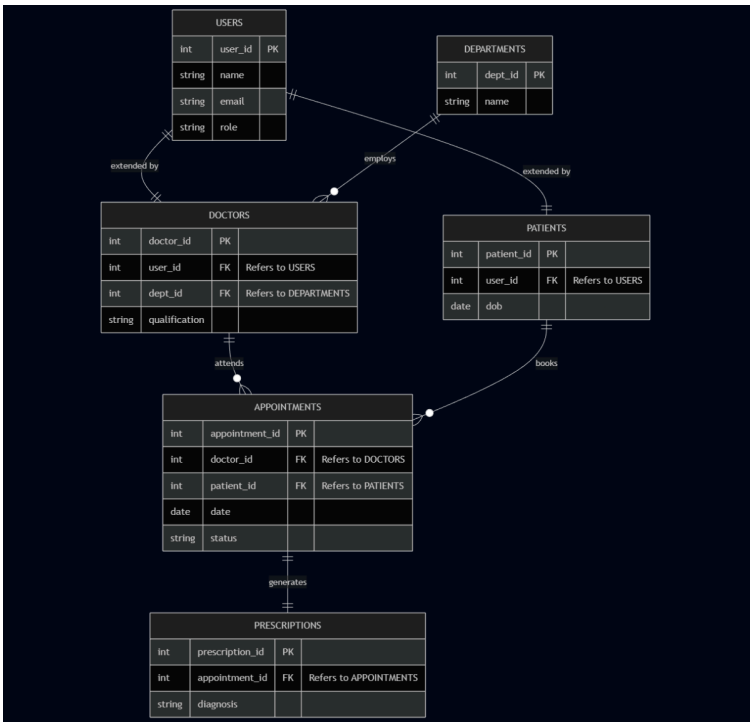
Category	Technology	Description
Backend	Flask, Flask-RESTful, Flask-SQLAlchemy, Flask-JWT-Extended, Flask-Caching, Celery	Provides a lightweight yet robust backend; Celery + Redis handle asynchronous tasks like emails and reports efficiently.
Frontend	Vue.js 3, Vite, Pinia, Bootstrap 5, Chart.js	Offers a responsive and reactive frontend experience.

Database	SQLite, Redis	SQLite serves as the relational database; Redis is used for Caching and as a Message Broker for Celery.

Database Schema Design

The schema is normalized to reduce redundancy and uses Foreign Keys to ensure data integrity. Indexes are placed on frequently queried fields (like `date`, `doctor_id`) for performance.

- **Users:** `user_id` (PK), `email`, `password`, `role` (Admin/Doctor/Patient), `status`. (Stores authentication and role info.)
- **Doctors:** `doctor_id` (PK), `user_id` (FK), `dept_id` (FK), `qualification`, `experience`. (Links to User and Department.)
- **Patients:** `patient_id` (PK), `user_id` (FK), `dob`, `address`. (Links to User.)
- **Departments:** `dept_id` (PK), `name`, `description`. (Categorizes doctors.)
- **Appointments:** `appointment_id` (PK), `doctor_id` (FK), `patient_id` (FK), `date`, `time`, `status`, `reason`. (Tracks visits.)
- **Prescriptions:** `id` (PK), `appointment_id` (FK), `diagnosis`, `medicine`. (Stores medical records.)



API Design

Authentication

- POST /api/auth/login - User Login (Returns JWT)
- POST /api/auth/register - User Registration

Admin

- GET /api/admin/stats - Dashboard Statistics (Counts & Charts)
- GET /api/departments - List all Departments
- POST /api/departments - Create new Department
- DELETE /api/departments/<id> - Delete Department

Doctors

- GET /api/doctors - List all Doctors
- POST /api/doctors - Create new Doctor (Admin only)
- GET /api/doctors/<id> - Get Doctor Details
- GET /api/doctor/availability - Get Doctor's Schedule
- POST /api/doctor/slots - Update Availability Slots

Patients

- GET /api/patients - List all Patients
- GET /api/patients/<id> - Get Patient Profile
- POST /api/export/treatments - Trigger CSV Export Job

Appointments

- GET /api/appointments - List Appointments (Filtered by Role)
- POST /api/appointments - Book Appointment
- PUT /api/appointments/<id> - Update Status (Cancel/Complete)
- GET /api/treatments/<appt_id> - Get Prescription Details
- POST /api/treatments/<appt_id> - Add Prescription (Doctor only)

Architecture and Features

The project follows a decoupled client-server architecture.

- **Backend (`backend/app`):** Structured with `models.py` for DB schema, `resources.py` for API logic, and `tasks.py` for background jobs.
- **Frontend (`frontend/src`):** Uses a component-based architecture with Vue.js, where `pages/` correspond to views and `services/api.js` handles HTTP requests.

Features ImplementedCore

- User Registration/Login
- Role-Based Dashboards (Admin/Doctor/Patient)
- Appointment Booking/Cancellation

Admin

- Manage Doctors/Departments
- View Statistics (Charts)
- Disable Users

Doctor

- Manage Schedule
- Treat Patients (Prescriptions)
- View History

Advanced

- **Background Jobs:** Daily Reminders, Monthly Reports (via Google Chat), CSV Export (Implemented via Celery).
- **Caching:** Redis caching for Departments/Doctors lists to improve performance.
- **Security:** JWT Authentication, Role-Based Access Control.

Video

Link:
https://drive.google.com/drive/folders/1vN-wvO9u8l86bDNYb2onvfD8wqsScgws?usp=sharing