App Dev Project Report

1. Student Details

Name: Akash Kumar Prasad **Roll Number:** 23f3000442
Email:23f3000442@ds.study.iitm.ac.in
About Me:
Hi! I'm Akash Kumar Prasad, a student currently pursuing the Diploma level of the IIT Madras Online BSc Degree in Programming and Data Science. I have a strong interest in technology and enjoy exploring how digital tools, software, and data can be used to solve real-world problems.

I'm continuously learning and improving my skills, and I enjoy taking on new challenges that help me grow both academically and personally. With a curious mindset and a passion for tech, I aim to build a solid foundation that will support my future career in the technology and data-driven world.

---

2. Project Details

Project Title: Hospital Management System (HMS)

Problem Statement:

To design and build a comprehensive web-based Hospital Management System that enables efficient management of hospital operations including patient registration, doctor management, appointment scheduling, treatment records, and administrative oversight. The system aims to streamline healthcare workflows by providing role-based dashboards for administrators, doctors, and patients.

Approach:

The application was built using a modern full-stack architecture with Flask as the backend RESTful API framework and Vue.js 3 as the Single Page Application (SPA) frontend. The system implements:

Role-based access control (Admin, Doctor, Patient) with JWT authentication

Modular backend architecture using Flask Blueprints for separation of concerns

Component-based frontend design with Vue Router for navigation

Redis caching for frequently accessed data (dashboard statistics, departments)

Celery background jobs for asynchronous operations (CSV exports, daily reminders, monthly reports)

SQLite database for persistent storage with SQLAlchemy ORM

---

3. AI/LLM Declaration

I used ChatGPT (GPT-5) for:

- SQLAlchemy boilerplate and relationship code
- Error handling suggestions for API routes
- Bootstrap class formatting
- Python docstring generation

AI usage was around 10–12%, limited to code scaffolding and syntax help. All core logic, database design, API architecture, frontend structure, authentication, and debugging were done by me. AI was used like Stack Overflow—all suggestions were reviewed and modified before use.

---

4. Technologies and Frameworks Used

Flask 3.0
Core backend web framework for building RESTful APIs

Flask-SQLAlchemy 3.1.1
Object Relational Mapper used with the SQLite database

Flask-JWT-Extended 4.6.0
JWT based authentication and session management

Flask-CORS 4.0.0
Enables cross origin communication between frontend and backend

Vue.js 3.5
Reactive frontend framework for Single Page Applications

Vue Router 4.6
Client side routing for SPA navigation

Axios 1.13
HTTP client for communicating with backend APIs

Bootstrap 5.3
Frontend styling framework with responsive components

Bootstrap Icons 1.13
Icon set used in UI components

Vite 7.2
Development server and build tool for the frontend

SQLite
Lightweight embedded database for local data storage

Redis 5.0
In memory cache and message broker used by Celery

Celery 5.3
Distributed task queue for background and asynchronous jobs

Werkzeug 3.0
Utilities for password hashing and security features

---

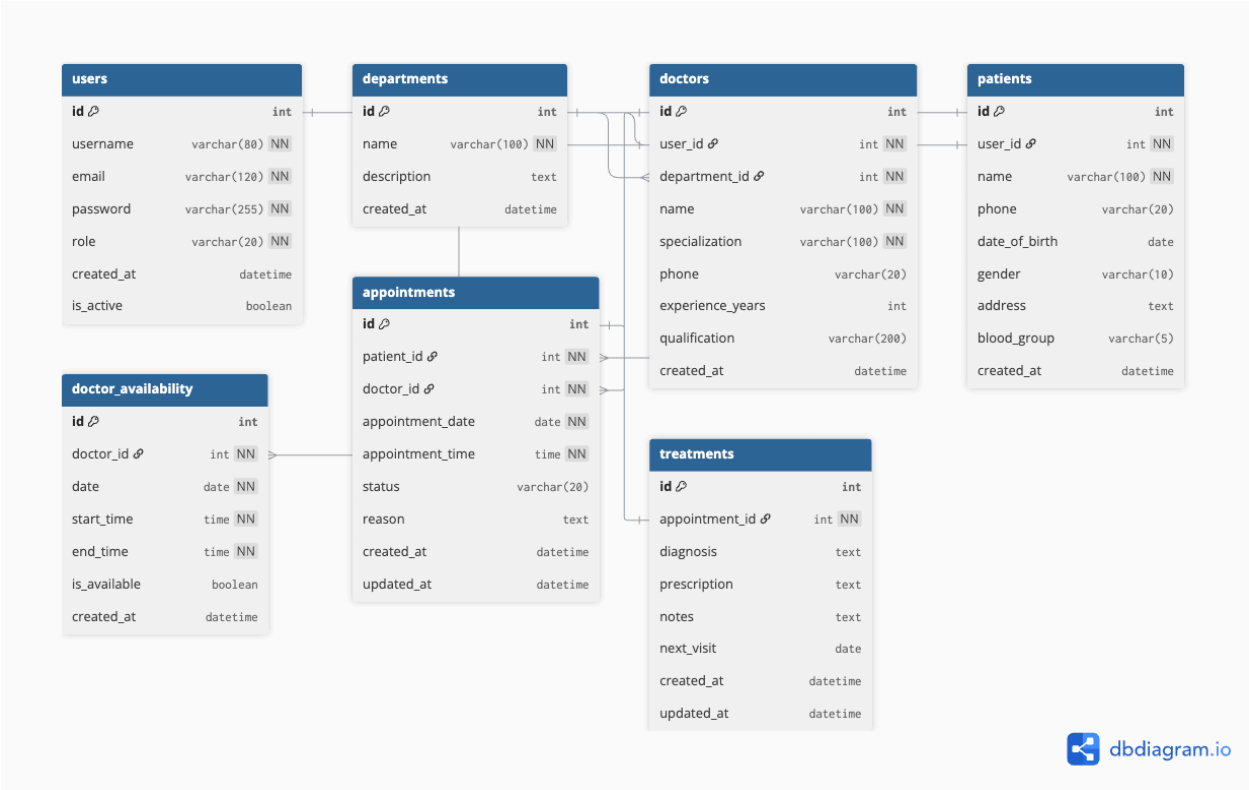
5. Database Schema / ER Diagram

Tables:

Table	Description	Key Fields
users	**User authentication and credentials**	**id, username, email, password, role (admin/doctor/patient), is_active, created_at**
departments	**Medical departments/specializations**	**id, name, description, created_at**
doctors	**Doctor profiles linked to users**	**id, user_id (FK), department_id (FK), name, specialization, phone, experience_years, qualification**
patients	**Patient profiles linked to users**	**id, user_id (FK), name, phone, date_of_birth, gender, address, blood_group**
doctor_availability	**Doctor scheduling slots**	**id, doctor_id (FK), date, start_time, end_time, is_available**
appointments	**Booking and scheduling records**	**id, patient_id (FK), doctor_id (FK), appointment_date, appointment_time, status, reason**
treatments	**Medical records and prescriptions**	**id, appointment_id (FK), diagnosis, prescription, notes, next_visit**

Relationships:

- One-to-One: User → Doctor (user_id)
- One-to-One: User → Patient (user_id)
- One-to-Many: Department → Doctors (department_id)
- One-to-Many: Doctor → Appointments (doctor_id)
- One-to-Many: Patient → Appointments (patient_id)
- One-to-Many: Doctor → DoctorAvailability (doctor_id)
- One-to-One: Appointment → Treatment (appointment_id)



---

6. API Resource Endpoints

Authentication Endpoints

Endpoint
Method
Description

`/api/auth/register`
POST
Register a new patient account

`/api/auth/login`
POST
Authenticate user and generate JWT token

`/api/auth/me`
GET
Retrieve current authenticated user information

Admin Endpoints

Endpoint
Method
Description

`/api/admin/dashboard`
GET
Dashboard statistics (cached)

`/api/admin/departments`
GET
List all departments (cached)

`/api/admin/doctors`
GET
List all doctors with search

`/api/admin/doctors`
POST
Create new doctor account

`/api/admin/doctors/:id`
PUT
Update doctor details

`/api/admin/doctors/:id`
DELETE
Deactivate or blacklist doctor

`/api/admin/patients`
GET
List all patients with search

`/api/admin/patients/:id`
PUT
Update patient details

`/api/admin/patients/:id`
DELETE
Deactivate or blacklist patient

`/api/admin/appointments`
GET
List all appointments with filters

`/api/admin/search`
GET
Search doctors or patients

Doctor Endpoints

Endpoint
Method
Description

`/api/doctor/dashboard`
GET
Doctor dashboard with statistics

`/api/doctor/appointments`
GET
List doctor’s appointments

`/api/doctor/appointments/:id/complete`
PUT
Mark appointment as completed with treatment details

`/api/doctor/appointments/:id/cancel`
PUT
Cancel an appointment

`/api/doctor/patients`
GET
List patients assigned to the doctor

`/api/doctor/patients/:id/history`
GET
Get treatment history of a patient

`/api/doctor/availability`
GET
Get doctor’s availability

`/api/doctor/availability`
POST
Set doctor availability for the next 7 days

Patient Endpoints

Endpoint
Method
Description

`/api/patient/dashboard`
GET
Patient dashboard data

`/api/patient/profile`
GET
Retrieve patient profile

`/api/patient/profile`
PUT
Update patient profile

`/api/patient/doctors`
GET
Search doctors with availability

`/api/patient/departments`
GET
List departments (cached)

`/api/patient/appointments`
POST
Book a new appointment

`/api/patient/appointments/:id`
PUT
Reschedule an appointment

`/api/patient/appointments/:id`
DELETE
Cancel an appointment

`/api/patient/treatment-history`
GET
Retrieve treatment history

`/api/patient/export-treatments`
GET
Export treatments as CSV (sync)

`/api/patient/export-treatments/async`
POST
Trigger asynchronous CSV export

`/api/patient/export-treatments/status/:task_id`
GET
Check asynchronous export status

7. Architecture and Features (optional)

hospitalmgmt-appdev2/

```
├── backend/
│   ├── app.py              # Main Flask application factory
│   ├── celery_tasks.py     # Celery background jobs & beat schedule
│   ├── requirements.txt    # Python dependencies
│   ├── config/
│   │   └── config.py       # App configuration (DB, JWT, Redis, Celery)
│   ├── models/
│   │   └── __init__.py     # SQLAlchemy database models
│   └── routes/
│       ├── auth.py         # Authentication routes (login, register)
│       ├── admin.py        # Admin routes with role decorator
│       ├── doctor.py       # Doctor routes with role decorator
│       └── patient.py      # Patient routes with role decorator
```

```
├── frontend/
│   ├── index.html            # Main HTML entry point
│   ├── package.json          # Node.js dependencies
│   ├── vite.config.js        # Vite build configuration
│   └── src/
│       ├── App.vue           # Root Vue component
│       ├── main.js           # Vue app initialization
│       ├── router/
│       │   └── index.js      # Vue Router configuration
│       ├── services/
│       │   └── api.js        # Axios API client with interceptors
│       ├── components/
│       │   ├── admin/        # Admin components (DoctorManagement, etc.)
│       │   ├── doctor/       # Doctor components (Appointments, etc.)
│       │   ├── patient/      # Patient components (DoctorsTab, etc.)
│       │   └── shared/       # Shared components (Navbar, Sidebar)
│       └── views/
│           ├── Login.vue
│           ├── Register.vue
│           ├── admin/AdminDashboard.vue
│           ├── doctor/DoctorDashboard.vue
│           └── patient/PatientDashboard.vue
└── README.md
```

Implemented Features:

Core Features:

- User registration and login with JWT authentication
- Role-based access control (Admin, Doctor, Patient)
- Protected routes with middleware decorators
- Secure password hashing with Werkzeug

Admin Features:

- Dashboard with statistics (doctors, patients, appointments)
- Create, update, deactivate doctors
- View, search, and deactivate patients
- View all appointments with status filtering
- Global search for doctors and patients

Doctor Features:

- Dashboard with upcoming appointments and patient stats
- View and manage appointments
- Mark appointments as completed with treatment details
- Add diagnosis, prescription, and notes
- View patient list and treatment history
- Set availability for next 7 days

Patient Features:

- Browse departments and search doctors
- View doctor availability slots
- Book, reschedule, and cancel appointments
- View treatment history with prescriptions
- Update profile information

Additional Features (Optional/Configured)

Redis Caching:

- Dashboard statistics cached with configurable timeout
- Department list cached for faster retrieval
- Cache invalidation on data modifications

Celery Background Jobs:

- Async CSV export with status polling API
- Daily appointment reminders (08:00 UTC) via Google Chat webhook
- Monthly doctor activity reports (1st of month) via email

Data Export:

- Export treatment history as CSV (synchronous)
- Async export via Celery with task status tracking

---

8. Video Presentation

Drive Link:
23f3000442AppDev2.mov
