HMS Summary Report

About Developer

Name: Ayush Kumar
Roll Number: 24f1001026
Email: 24f1001026@ds.study.iitm.ac.in and kumarayush0883professional@gmail.com

My name is **Ayush Kumar Singh**, and I am originally from **Varanasi**. I am a learner of **IITM BS** program from **2024**. Currently I am at diploma level . It's my 3rd term .I am also pursuing my **B.Sc Mathematics** from **University of Allahabad.** I have a growing interest in software development, data science, and problem-solving.I like working on Python, web development, and mathematical logic, and I am slowly expanding my skills in these areas.

About Project

According to the project statement we have to build a Hospital management system for a Hospital , which can manage all the work of the hospital instead of manual methods .

Project 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 :
First I have understood about users and their roles and their features like for admin , he is pre define and also can block, delete and create doctors and patients .
Similarly for doctor and patient

Then create their DB model and required table and then go create route for implementing functionality like login , registration and many more ……

And after that creating routes I created html files in templates folder to render that info in frontend from backend ..

Technologies and Frameworks Used

Technology / Library	Purpose
Python	For Writing routes and all codes

Flask	Core backend web framework
SQLAlchemy	Object Relational Mapper for SQLite database
Jinja2	Template engine for rendering dynamic HTML pages
Bootstrap 5	Frontend styling and responsive design
SQLite	Lightweight local database for storing user data
Werkzeug	Used for password hashing and verification
JSON Library	Used for handling doctor availability
datetime	You use it for: Time slot validation Checking appointment times

Database Schema / ER Diagram

Admin class

1. **id:** integer, primary key
2. **username:** string, unique, cannot have null value
3. **email:** string, unique, cannot have null value
4. **mobile:** string, unique, cannot have null value
5. **password:** string, cannot have null value
6. **role:** string, default = "admin", cannot have null value
7. **Relationship:**
    - One Admin can create many Doctors
    - One Admin can create/manage many Departments

Doctor class

1. **id:** integer, primary key
2. **username:** string, unique, cannot have null value
3. **email:** string, unique, cannot have null value
4. **mobile:** string, unique, cannot have null value
5. **password:** string, cannot have null value
6. **department:** string, can have null value
7. **availability:** string (JSON), can have null value
8. **status:** string, default = "active", cannot have null value
9. **role:** string, default = "doctor", cannot have null value

10. **created_by_admin_id:** foreign key referencing Admin
11. **Relationship:**
    - One Doctor has many Appointments
    - One Doctor handles many Treatments

3. Patient class

1. **id:** integer, primary key
2. **username:** string, unique, cannot have null value
3. **email:** string, unique, can have null value
4. **mobile:** string, unique, cannot have null value
5. **password:** string, cannot have null value
6. **status:** string, default = "active", cannot have null value
7. **role:** string, default = "patient", cannot have null value
8. **Relationship:**
    - One Patient can have many Appointments
    - One Patient can have many Treatments

Appointment class

1. **appointment_id:** integer, primary key
2. **patient_id:** foreign key referencing Patient
3. **doctor_id:** foreign key referencing Doctor
4. **appointment_date:** date, cannot have null value
5. **appointment_time:** time, cannot have null value
6. **reason_for_visit:** string, can have null value
7. **appointment_status:** string, default = "Booked", cannot have null value
8. **Relationship:**
    - One Appointment has one Treatment

Treatment class

1. **treatment_id:** integer, primary key
2. **appointment_id:** foreign key referencing Appointment
3. **diagnosis:** string, cannot have null value
4. **prescription:** string, cannot have null value
5. **treatment_date:** date, cannot have null value
6. **follow_up_date:** date, can have null value
7. **notes:** string, can have null value
8. **doctor_id:** foreign key referencing Doctor

9. **patient_id:** foreign key referencing Patient

Department class

1. **department_id:** integer, primary key
2. **department_name:** string, unique, cannot have null value
3. **department_description:** string, cannot have null value
4. **created_by_admin_id:** foreign key referencing Admin
5. **Relationship:**
    - One Department is created/managed by one Admin

All Relationships

1. **Admin → Doctor**
    - One Admin can create **many Doctors**
    - Relationship: **1-to-Many**

2. **Admin → Department**
    - One Admin can create/manage **many Departments**
    - Relationship: **1-to-Many**

3. **Doctor → Appointment**
    - One Doctor can have **many Appointments**
    - Relationship: **1-to-Many**

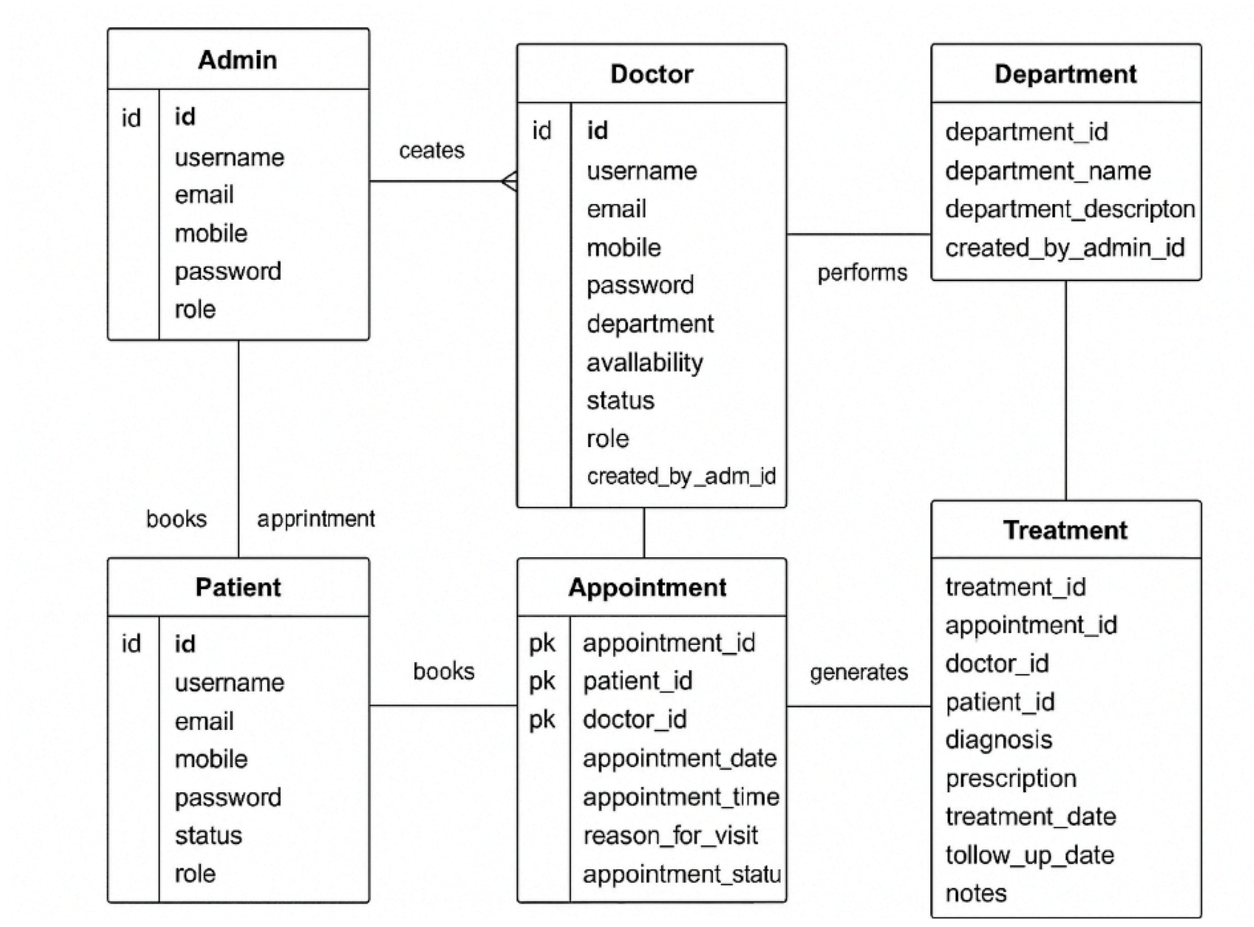
4. **Patient → Appointment**
    - One Patient can have **many Appointments**
    - Relationship: **1-to-Many**

5. **Appointment → Treatment**
    - One Appointment has **one Treatment**
    - Relationship: **1-to-1**

6. **Doctor → Treatment**
    - One Doctor can handle **many Treatments**

    ○ Relationship: **1-to-Many**

7. **Patient → Treatment**
    ○ One Patient can have **many Treatments**
    ○ Relationship: **1-to-Many**



Architecture and Feature

Architecture Overview:

- **app.py** – main Flask application entry point containing route for all functionality
- **models.py** – database models using SQLAlchemy
- **/templates** – containing all required Jinja2 HTML templates
- **/instance -** folder which contains database

Implemented Features:

- Login for all users
- Registration only for patients
- CRUD Operations for Admin
- Update availability feature for doctor
- Make appointment for patient
- Search for admin for doctor and patient by their name
- Mark appointment as complete (for doctor) or cancel for all users
- Search appointment for all users for searching appointment by their name and date

AI/LLM Declaration

I used **ChatGPT (GPT-5)** to assist in styling and frontend (because I only know basic CSS) . and also for creating search functionality (all users), update availability (for doctors) and booking appointment (for patient) routes, I used AI for understanding logic and implementing it

Video Presentation

Here is Video Presentation :
https://drive.google.com/file/d/1fgPzBZOYG8KPz67iGCCJ6tY3oybWCIUX/view?usp=sharing

Note : Pardon me for Treatment (for doctor)and patient history (for all) Functionality is not implemented till submission Today is last day of submission so I will fix it after final viva .