HealNest

App Development Project Report

1. Student Details

Name: Ashish Vishwakarma

Roll Number: 23F3000163

Email: 23f3000163@ds.study.iitm.ac.in

About Me: I am a student in the IIT Madras BS Degree program with a keen interest in full-stack web development. I enjoy building practical applications that solve real-world administrative and logistical problems using Python and modern web frameworks.

2. Project Details

Project Title: HealNest - Hospital Management System

Problem Statement: Traditional hospital management often suffers from manual inefficiencies, such as double-booked appointments, lost patient history records, and lack of visibility into doctor availability. There is a need for a centralized, digital platform to streamline operations between administrators, doctors, and patients.

Approach: HealNest is a web-based application built using the Flask framework (MVC architecture). It solves these problems by implementing Role-Based Access Control (RBAC) for three distinct user types. It utilizes a relational database to link patients, doctors, and appointments, ensuring data integrity and real-time scheduling updates.

3. AI/LLM Declaration

I used Large Language Models (e.g., ChatGPT/Gemini) to assist in the development process.

- **Usage:** Approximately 30%
- **Purpose:** The AI assistance mainly helped with generating boilerplate HTML/CSS templates, resolving SQLAlchemy relationship issues, and refining JavaScript for features like the appointment time-slot picker and PDF export. It also aided in improving documentation clarity and formatting.
- **Statement:** All core implementation logic, system design decisions, debugging, testing, and deployment tasks were performed and verified manually to ensure correctness and originality.

4. Technologies and Frameworks Used

Technology / Library	Purpose
Technology	**Purpose**
Flask	Core backend web framework
SQLAlchemy	ORM for managing the SQLite database
SQLite	Local relational database for storing users and records
Flask-Login	User authentication and session management
Flask-WTF	Secure form handling and validation
Bootstrap 5	Responsive frontend styling and UI components
Chart.js	Visualizing hospital statistics (Admin Dashboard)
JavaScript	Dynamic frontend logic (PDF generation, async requests)

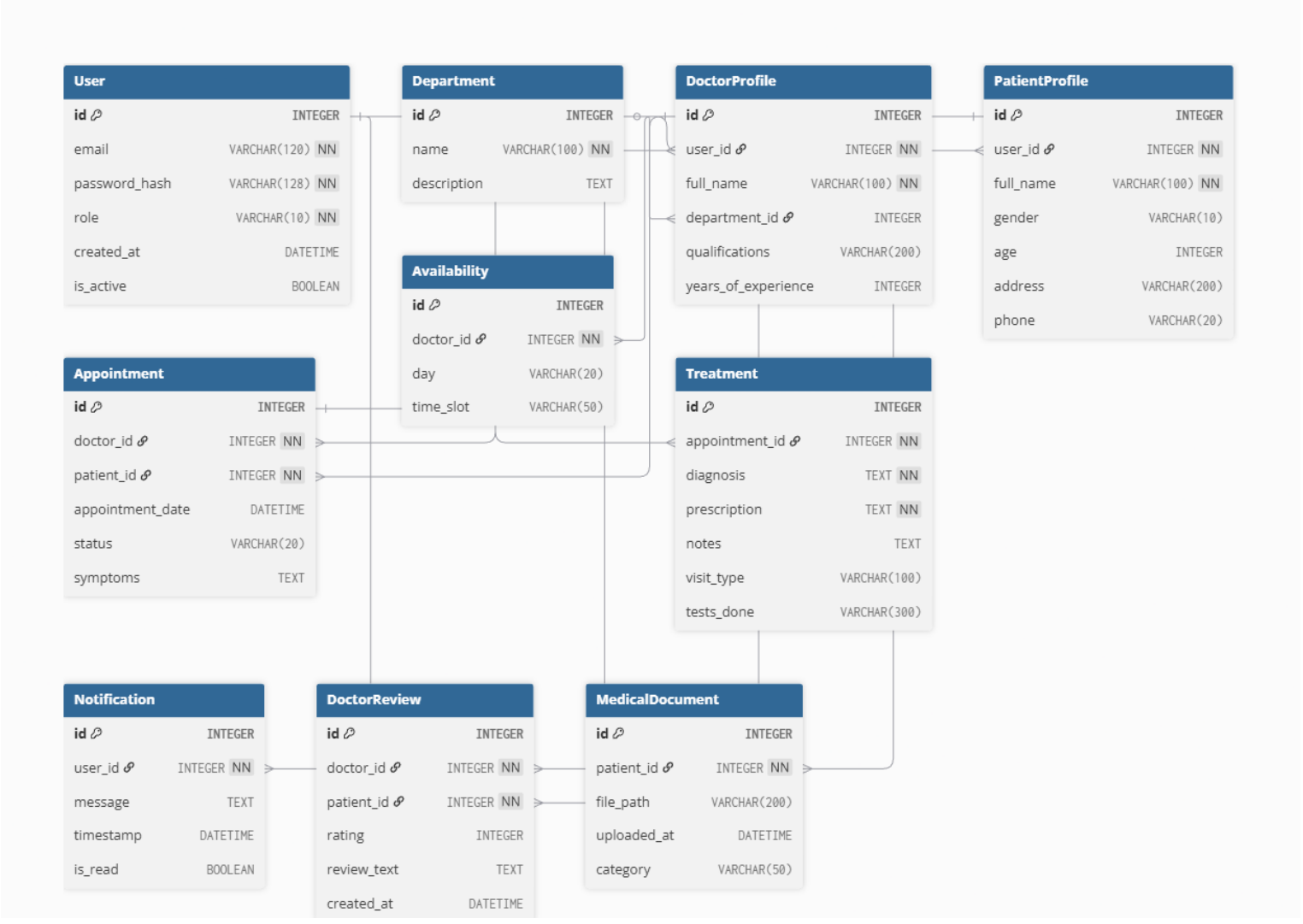
5. Database Schema / ER Diagram

The application uses a relational schema with the following core tables:

- **User: Central authentication table (Admin, Doctor, Patient) with an is_active flag for blacklisting.**
- **Profiles: DoctorProfile and PatientProfile linked 1-to-1 with User for role-specific data.**
- **Department: Categorizes doctors (e.g., Cardiology, Neurology).**
- **Appointment: Links Patients to Doctors at specific times.**
- **Availability: Stores doctor's weekly schedule slots.**
- **Treatment: Stores medical history (Diagnosis, Prescription) for completed appointments.**

Key Relationships:

- **User 1—1 Profile**
- **Department 1—N Doctors**
- **Doctor/Patient 1—N Appointments**



6. API Resource Endpoints

Endpoint	Method	Description
`/login`	**POST**	Authenticates user credentials and redirects to the appropriate dashboard based on role (Admin, Doctor, Patient).
`/admin/api/dashboard-stats`	**GET**	Returns analytics data including total users, appointments, and department distribution for admin charts.
`/patient/book/<id>`	**POST**	Books an appointment with a doctor while preventing slot conflicts and enforcing availability validation.
`/doctor/availability`	**POST**	Saves or updates the doctor's weekly schedule, making slots visible for booking.
`/doctor/treat/<id>`	**POST**	Updates appointment status and stores medical records including diagnosis and prescription.
`/patient/appointments`	**GET**	Retrieves all appointments (upcoming and completed) for the logged-in patient.

7. Architecture and Features

Architecture Overview:

The app follows a modular structure using Flask Blueprints (admin_bp, doctor_bp, patient_bp) to separate logic. It uses Jinja2 templating for server-side rendering, enhanced with client-side JavaScript for interactivity.

Key Features:

- Role-Based Dashboards: Distinct UIs for Admins, Doctors, and Patients.
- Smart Scheduling: Dynamic booking system that prevents conflicts based on doctor availability.

- Medical History: Centralized, read-only history view for all roles with PDF export functionality.
- Admin Analytics: Visual charts showing registration trends and department activity.
- Access Control: Admin capability to blacklist/suspend users (Soft Delete).

8. Video Presentation

Drive Link:
https://drive.google.com/file/d/1vonoOzLdEwde9PSYq2Hz5J5G1xm_ZvZ4/view?usp=sharing