App Development Project Report

App Dev Project Report

1. Student Details

Name: Shri Krishna Pandey
Roll Number: BS22XXXX
Email: skpandey@onlinedegree.iitm.ac.in
About Me: I am a Course Instructor at IIT Madras BS Degree program with a deep interest in web application development and data-driven technologies. I enjoy building meaningful applications that combine learning, analytics, and user experience.

---

2. Project Details

Project Title: Quantified Self App

Problem Statement:

To design and 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 app was built using Flask as the backend framework with a modular structure. It allows users to view their past existing and future appointments. It also allows them to book future appointments. It allows the doctors to view their appointments and set their availability. It provides a super user access to an admin who can view all the users, search for particular users and add doctors and blacklist them if required.

3. AI/LLM Declaration

I used **Google Gemini (3 Pro)** to assist in checking the routes for errors and fixing them. It was also used to assist with the html templates and modify the styling of the templates created

The extent of AI/LLM usage is around **10–15%**, 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
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

---

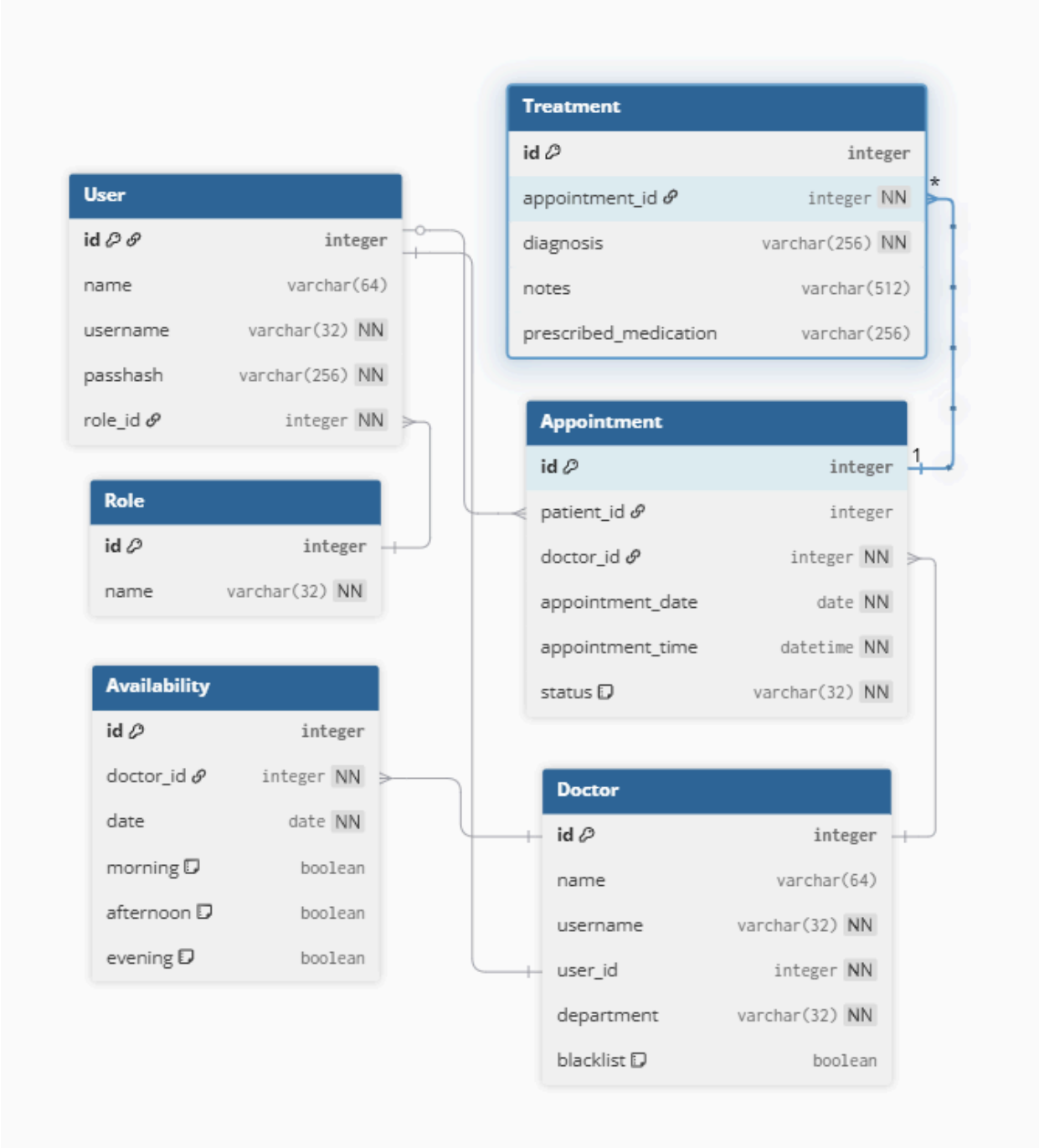
5. Database Schema / ER Diagram

Tables:

1. **User** — stores user profile details (id, name, username, passhash)
2. **Role** — stores user permissions (id, name)
3. **Doctor** — stores specific doctor profile details linked to a user account (id, name, username, user_id, department, blacklist)
4. **Appointment** — stores booking details between a patient and a doctor (id, name, username, user_id, department, blacklist)
5. **Treatment** — stores medical diagnosis and prescription for a specific appointment (id, appointment_id, diagnosis, notes, prescribed_medication)
6. **Availability** — stores a doctor's schedule status for specific dates (id, doctor_id, date, morning, afternoon , night)

Relationships:

- One-to-Many → `Role→ User`
- One-to-One → `User → Doctor`
- One-to-Many → `User(Patient)→ Appointment`
- One-to-Many → `Doctor→ Appointment`
- One-to-Many → `Doctor→ Availability`
- One-to-Many → `Appointment→ Treatment`



(made using dbdiagram.io)

---

6. API Resource Endpoints

No APIs used

---

7. Architecture and Features (optional)

Architecture Overview:

- **app.py** – main Flask application entry point
- **/templates** – Jinja2 HTML templates
- **/static** – CSS, JS, and chart visualization files

Implemented Features:

- User registration and login.
- Role based access control system using a decorator.
- Admin can add, update, remove and blacklist doctors. Additionally they can view and remove patients.
- An availability setting system for doctors to set their availability for 7 days.
- Secure password hashing and user session handling.
- Appointment booking system for patients and management system for doctors.
- Consultation feedback system and medical history tracking.
- Admin can search for specific patients and doctors by name, username or department.

---

8. Video Presentation

Drive Link:
https://drive.google.com/file/d/1uJPZbUKeirYXpEMs0hKdR-qnme3XoNEz/view?usp=sharing

---