

Author

Name: Kumuda Sri Padmanabhuni

Roll Number: 24f3000060

Email: 24f3000060@ds.study.iitm.ac.in

About Me: I am a Dual Degree Student in Computer Science and a Tech Enthusiast, passionate about Data Science, Web development, Problem solving and always excited to learn new things.

Description

LilacBridge is a Flask-based hospital management system that centralizes hospital operations for admin, doctors, and patients by managing doctors, availability, and appointments in one place. Patients search by specialization, book/reschedule/cancel within doctor availability, and view treatment history, while doctors record diagnosis, prescriptions, and notes and admin manages users and full appointment logs.

AI/LLM Usage

I used ChatGPT (GPT-5) to assist in writing Bootstrap and CSS(for styling and layouts), and form and Validations (using JS and Flask-WTF).

The extent of AI/LLM usage is around **10-12%** limited to code suggestions only.

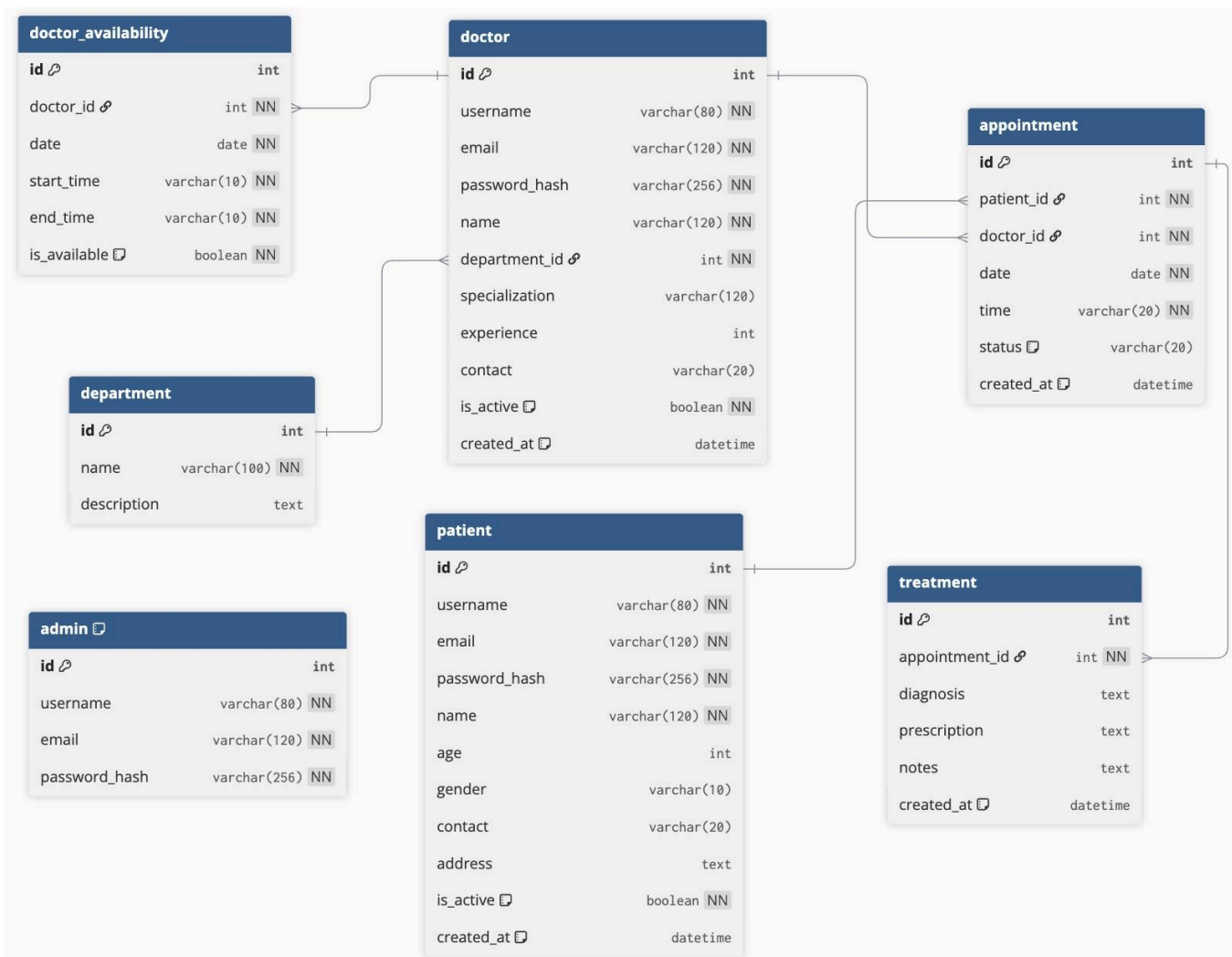
All final implementation logic, debugging and integration were done on my own manually.

Technologies used

- **Flask** - Lightweight web framework for handling routes and rendering templates.
- **Flask-Login** - Manages user authentication and session handling.
- **SQLAlchemy** - Object Relational Mapper for SQLite database
- **Jinja2** - Template engine for rendering dynamic HTML pages
- **Bootstrap 5** - Frontend styling and responsive design
- **Werkzeug Security** - Provides password hashing for secure authentication.
- **Datetime** - Handles date and time operations efficiently.
- **WTForms** - Frontend form validation
- **SQLite** - Lightweight local database for storing user data

DB Schema Design

- **Department → Doctor:**
One-to-many where each department can have multiple doctors via doctor.department_id.
- **Doctor → DoctorAvailability:**
One-to-many where each doctor defines multiple date/time windows.
- **Patient → Appointment:**
One-to-many where a patient can book many appointments.
- **Doctor → Appointment:**
One-to-many where a doctor can receive many appointments.
- **Appointment → Treatment:**
One-to-one where a completed appointment has exactly one treatment with diagnosis, prescription, and notes.
- **Admin** is a standalone authentication entity with no foreign keys to the other entities.



API Design

- **Authentication:** /login and /register — sign in or sign up Admin, Doctor, and Patient users, establishing role-based sessions.
- **Departments & Doctors:** /patient/department/<dept_id> — list active doctors in a department; admin adds/edits/activates/blacklists doctors.
- **Availability:** /doctor/availability — doctors set date/time windows; slots are derived in 30-minute steps (06:00–23:30).
- **Appointments:**
 - **Book:** /patient/book_appointment/<doctor_id> — weekly slot grid from availability minus booked/past times.
 - **Cancel:** /patient/cancel_appointment/<appointment_id>
 - **Reschedule:** /patient/reschedule_appointment/<appointment_id> — update to a new valid slot, freeing the old one.
- **Treatments:** /doctor/complete_appointment/<appointment_id> — save diagnosis, prescription, notes; visible in patient history and admin registry.
- **Dashboards:**
 - **Admin:** /admin/dashboard and /admin/appointments — shows totals for doctors, patients, and appointments; includes search; lists doctors/patients with Edit and Disable/Activate; and shows appointment details.
 - **Doctor:** /doctor/dashboard, /doctor/appointments — upcoming and past appointments.

- **Patient:** /patient/dashboard, /patient/appointment_history, /patient/doctor_profile/<doctor_id> — discover, book, and view treatment history.

*Endpoints are implemented as Flask routes returning rendered Jinja2 templates.

Architecture and Features

The project follows an MVC (Model-View-Controller) structure. **Controllers** are in app.py handling routes and logic, **Models** (database interactions) are managed via SQLAlchemy in models.py. **Templates** (Jinja2 HTML template files) are in the templates/ folder. Static assets like css and logo of the app are stored in the static/ folder.

Features implemented:

- **Core Features:** Secure authentication, role-specific dashboards, availability-aware booking, cancel/reschedule flows, and programmatic DB initialization.
- **Admin Features:** Add/edit/activate/blacklist doctors and patients, global search, total number of doctors/patients/appointments stats, and full appointment registry with treatment details.
- **Doctor Features:** Define availability (06:00–23:30), view upcoming appointments, complete visits with diagnosis/prescription/notes, and view patient history.
- **Patient Features:** Register/login/update profile, browse by department/specialization, book within available slots, reschedule/cancel, and view completed treatment details.
- **Additional Features:** Bootstrap-enhanced UI, server-side slot validation, and hashed passwords for all users.

Video

https://drive.google.com/file/d/1MIZmiNMt8m-kqUVQ8Sp08wThwDO9wayA/view?usp=drive_link