App Dev Project Report

1. Student Details

Name: Atharva Thakre

Roll Number: 24F1001859

Email: 24f1001859@ds.study.iitm.ac.in

About Me: I am a tech enthusiast currently studying Computer Science and Engineering, while also diving into Data Science on the side. I love building cool things with Java, Python, and modern web tech, whether it's a touchless gesture-based control system or tools that make everyday work easier.

---

2. Project Details

Project Title: Hospital Management System

Problem Statement:
To design and build a comprehensive web-based hospital management application that enables efficient management of doctors, patients, appointments, and medical records through role-based access control for admins, doctors, and patients.

Approach:
The application was built using Flask as the backend framework with a modular MVC architecture. It implements three distinct user interfaces based on roles: Admin (for hospital management), Doctor (for appointment and patient management), and Patient (for booking appointments and accessing medical history). The system uses SQLAlchemy ORM for database operations and implements RESTful APIs for external integration.

---

3. AI/LLM Declaration

What AI was used for (Limited Scope):

- SQLAlchemy Models: Column definitions and basic relationship syntax
- HTML Templates: Bootstrap component structure (navbar, cards, forms)
- Flask Routes: Basic decorator patterns
- Form Processing: pattern suggestions
- Business logic implementation:
    - Role-based authorization decorator
- Security implementations:
    - Password hashing with PBKDF2-SHA256
    - Session management configuration
- Query optimization - Filtering, joins, ordering logic
- Template integration - Jinja2 logic, loops

What was completely MANUAL (No AI):

- Complete database schema design - All 7 tables, columns, relationships, constraints
- Business logic implementation:
    - Appointment booking conflict checking
    - Doctor availability slot management
    - Treatment record linking to appointments
- Security implementations:
    - User authentication flow
    - Role validation logic
- Feature integration:
    - Admin CRUD workflows
    - Doctor dashboard with appointment filtering
    - Patient self-registration and booking system
    - Search functionality with filters
- Database initialization:
    - Auto-creation of departments
    - Admin user seeding
    - Default data strategy
- All testing and debugging - Every feature tested manually

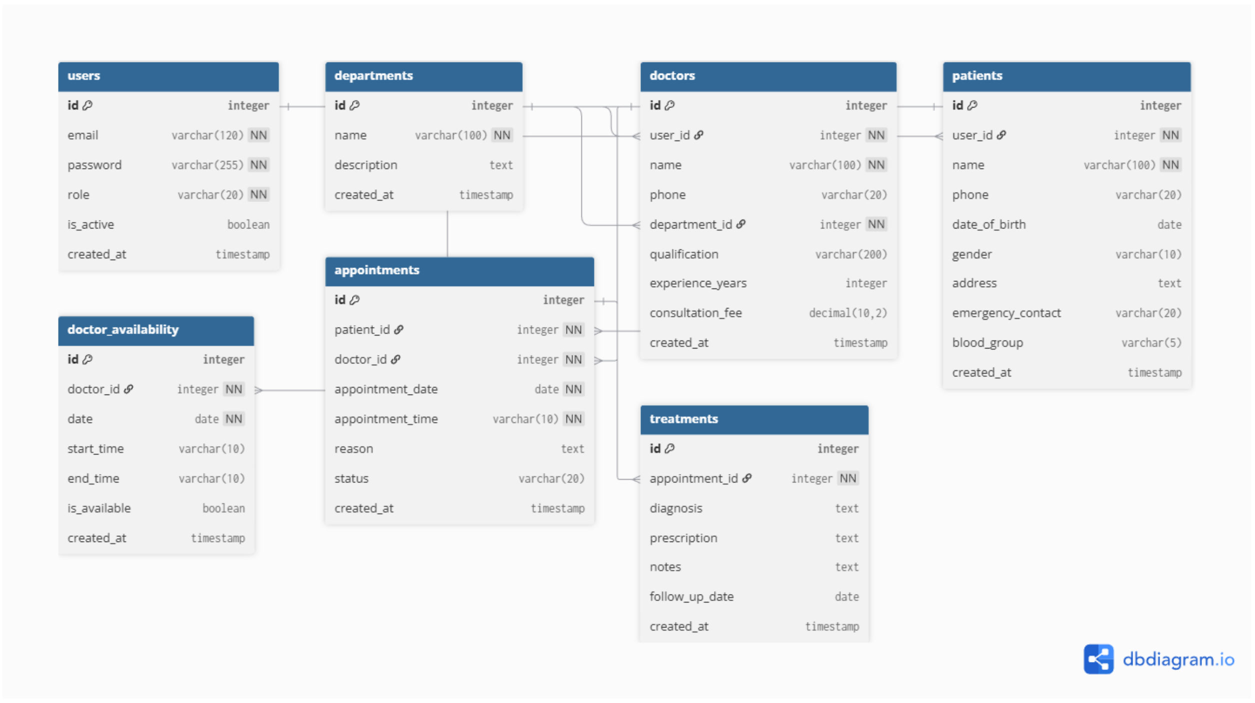
---

4. Technologies and Frameworks Used

Technology / Library	Purpose
Flask 2.3.0	Core backend web framework for routing and request handling
Flask-SQLAlchemy	Object Relational Mapper (ORM) for database operations
Flask-Login	User authentication, session management, and login protection
SQLite	Lightweight embedded database for data persistence
Jinja2	Template engine for rendering dynamic HTML with inheritance
Bootstrap 5.3	Frontend CSS framework for responsive UI design
Werkzeug	Password hashing (PBKDF2-SHA256) and security utilities

---

5. Database Schema / ER Diagram



---

6. API Resource EndpointsSample Response Examples:

Endpoint	Method	Description
/api/departments	GET	Fetch all departments with doctor count
/api/doctors	GET	Fetch doctors list (optional filter: ?department_id=X)
/api/doctor/<id>/availability	GET	Fetch 7-day availability schedule for specific doctor

---

7. Architecture and Features

Architecture Overview:

- app.py – main Flask application entry point
- /models – database models using SQLAlchemy
- /routes – Flask Blueprints for user and activity routes
- /templates – Jinja2 HTML templates
- /static – CSS, JS, and chart visualization files

Key Components:

- Controllers: All routes organized by role prefix (/admin/*, /doctor/*, /patient/*)
- Models: Separate Python files for each entity in models folder
- Views: Jinja2 templates with template inheritance from base.html
- Authentication: Flask-Login with session management and password hashing
- Authorization: Custom @role_required() decorator for route protection

Implemented Features:

1. User Authentication & Authorization
   - Secure login/logout with Flask-Login
   - Role-based access control (Admin, Doctor, Patient)
   - Password hashing using PBKDF2-SHA256
   - Session management and login protection
2. Admin Features
   - Dashboard with statistics (doctors, patients, appointments count)

    - Complete CRUD operations for doctors
    - Patient management with search functionality
    - Appointment oversight and monitoring
    - Search doctors by name/department
3. Doctor Features
    - Personal dashboard with today's appointments
    - Weekly appointment schedule view
    - Set 7-day availability (date, time slots)
    - Complete appointments with treatment records
    - View patient medical history
    - Patient list management
4. Patient Features
    - Self-registration with profile details
    - Search doctors by name and department
    - Filter doctors by department
    - Book appointments with slot availability check
    - View upcoming and past appointments
    - Cancel booked appointments
    - Access complete medical history with treatments

---

8. Video Demonstration

Video Link:
https://drive.google.com/file/d/1DrGmB81hyfgjMNkCXkcHGBSHFyGM5t5Y/view?usp=drive_link

---

9. Conclusion

The Hospital Management System successfully implements a complete role-based web application with comprehensive CRUD operations, RESTful APIs, and modern UI/UX design. The modular architecture ensures maintainability and scalability, while Flask-SQLAlchemy provides efficient database operations. The application meets all mandatory requirements and includes several optional enhancements for improved functionality and user experience.