Hospital Management System V2 Project Report

Student Name: Utkarsh Shukla ([21f2001497@ds.study.iitm.ac.in])
Student ID: 21f2001497
Course: MAD 2
Institution: IIT MADRAS
Submission Date: November 30, 2024
Project Title: Hospital Management System V2

Project Approach

1. **Role-Based Access Control:** Separate interfaces for Admins, Doctors, and Patients
2. **Automated Scheduling:** Smart appointment booking with availability management
3. **Centralized Data Management:** Unified database for all hospital operations
4. **Real-Time Notifications:** Automated reminders and reporting via Google Chat integration
5. **Modern Web Architecture:** Scalable and maintainable system using modern frameworks

Key Features Implemented

User Management:

- Multi-role authentication (Admin, Doctor, Patient)
- JWT-based security with token refresh
- Profile management for all user types

Appointment System:

- Doctor availability management with weekly schedules
- Patient appointment booking with conflict prevention
- Real-time status tracking (booked, completed, cancelled)
- Treatment record management

Automated Job Processing:

- Daily appointment reminders at 8:00 AM
- Monthly performance reports generation
- CSV export functionality for patient records
- Google Chat integration for notifications

Dashboard Features:

- **Admin Dashboard:** System statistics, user management, job monitoring
- **Doctor Dashboard:** Appointment management, patient records, availability settings

- **Patient Dashboard:** Doctor browsing, appointment history, profile management

AI/LLM Declaration

During the development of this Hospital Management System, I utilized **Gemini 2.5 Pro** as an AI assistant for the following specific purposes:

CSS Styling and Formatting Assistance:

- Received guidance on Bootstrap 5 component styling and responsive design principles
- Obtained suggestions for improving UI/UX design patterns and color schemes
- Got help with CSS Grid and Flexbox layouts for dashboard components
- Received recommendations for modern web design best practices

Learning and Understanding:

- Used AI to better understand Vue.js 3 Composition API concepts and best practices
- Gained insights into Flask-SQLAlchemy relationship configurations
- Learned about Celery task queue implementation patterns
- Understood JWT authentication flow and security considerations

Important Note: Including documentation also, my usage pretty clearly falls under <10% mark.

Frameworks and Libraries Used

- **Flask 3.0.0** - Python web framework for API development
- **Flask-SQLAlchemy 3.1.1** - ORM for database operations
- **Flask-JWT-Extended 4.5.3** - JWT authentication and authorization
- **Flask-CORS 4.0.0** - Cross-origin resource sharing support
- **SQLite** - Lightweight database for development and testing
- **Flask-Caching 2.1.0** - Redis-based caching for performance optimization
- **Redis 5.0.1** - In-memory data store for caching and message brokering
- **Celery 5.3.4** - Distributed task queue for background job processing
- **Redis** - Message broker for Celery task queue
- **Werkzeug 3.0.1** - WSGI utility library
- **Requests 2.31.0** - HTTP library for external API calls
- **Python-dotenv 1.0.0** - Environment variable management
- **Email-validator 2.1.0** - Email validation utilities
- **Vue.js 3.3.4** - Progressive JavaScript framework with Composition API
- **Vue Router 4.2.5** - Official router for Vue.js applications
- **Vuex 4.1.0** - State management pattern and library
- **Bootstrap 5.3.2** - CSS framework for responsive design
- **Custom CSS** - Additional styling for enhanced user experience
- **Vite 5.0.0** - Fast build tool and development server

- **Axios 1.6.0** - Promise-based HTTP client for API communication
- **@vitejs/plugin-vue 4.4.0** - Vue.js plugin for Vite
- **Node.js 16+** - JavaScript runtime for frontend development
- **Python 3.11+** - Backend runtime environment
- **npm** - Package manager for frontend dependencies
- **pip** - Package manager for Python dependencies

Database Tables Description

1. USERS Table

- Primary authentication table for all system users
- Stores email, hashed passwords, and role information
- Supports Admin, Doctor, and Patient roles
- Includes account status and creation timestamp

2. DEPARTMENTS Table

- Medical departments/specializations catalog
- Referenced by doctors for organizational structure
- Includes department descriptions

3. DOCTORS Table

- Doctor profile information linked to user accounts
- Stores professional details like specialization and experience
- Foreign key relationship with departments and users

4. PATIENTS Table

- Patient profile information and medical history
- Comprehensive demographic and health information
- Linked to user accounts for authentication

5. APPOINTMENTS Table

- Central appointment scheduling table
- Links patients and doctors with date/time slots
- Tracks appointment status throughout lifecycle
- Includes unique constraint on doctor_id, date, time

6. TREATMENTS Table

- Medical records for completed appointments
- Stores diagnosis, prescriptions, and treatment notes
- One-to-one relationship with appointments

7. AVAILABILITY Table

- Doctor availability schedule management
- Weekly recurring schedule with day-of-week mapping
- Supports multiple time slots per day per doctor

Authentication Endpoints

```
POST   /api/auth/register          - Patient registration
POST   /api/auth/login             - User authentication
POST   /api/auth/refresh           - Token refresh
GET    /api/auth/me                - Get current user info
POST   /api/auth/logout            - User logout
```

Admin Endpoints

```
GET    /api/admin/stats            - System statistics
GET    /api/admin/doctors          - List all doctors
POST   /api/admin/doctors          - Create new doctor
GET    /api/admin/doctors/{id}     - Get doctor details
PUT    /api/admin/doctors/{id}     - Update doctor
DELETE /api/admin/doctors/{id}     - Deactivate doctor
GET    /api/admin/patients         - List all patients
GET    /api/admin/patients/{id}    - Get patient details
PUT    /api/admin/patients/{id}    - Update patient
DELETE /api/admin/patients/{id}    - Deactivate patient
GET    /api/admin/appointments     - List all appointments
GET    /api/admin/appointments/upcoming - Upcoming appointments
GET    /api/admin/search           - Search doctors/patients
GET    /api/admin/departments      - List departments
PUT    /api/admin/users/{id}/toggle-status - Toggle user status
POST   /api/admin/jobs/trigger-reminders   - Trigger daily reminders
POST   /api/admin/jobs/trigger-reports     - Trigger monthly reports
GET    /api/admin/jobs/{task_id}/status    - Get job status
GET    /api/admin/jobs/active              - List active jobs
```

Doctor Endpoints

```
GET    /api/doctor/stats           - Doctor dashboard statistics
GET    /api/doctor/appointments/today      - Today's appointments
GET    /api/doctor/appointments/upcoming   - Upcoming appointments
GET    /api/doctor/appointments            - All appointments
PUT    /api/doctor/appointments/{id}/complete - Mark appointment complete
PUT    /api/doctor/appointments/{id}/cancel   - Cancel appointment
POST   /api/doctor/appointments/{id}/treatment - Add treatment record
PUT    /api/doctor/appointments/{id}/treatment - Update treatment
GET    /api/doctor/patients/{id}/history   - Patient treatment history
GET    /api/doctor/availability            - Get availability schedule
POST   /api/doctor/availability            - Add availability slot
DELETE /api/doctor/availability/{id}       - Remove availability slot
```

Patient Endpoints

```
GET    /api/patient/stats          - Patient dashboard statistics
GET    /api/patient/departments    - List medical departments
GET    /api/patient/doctors        - List available doctors
GET    /api/patient/doctors/{id}   - Get doctor details with availability
POST   /api/patient/appointments   - Book new appointment
GET    /api/patient/appointments/upcoming - Upcoming appointments
GET    /api/patient/appointments/past     - Past appointments
PUT    /api/patient/appointments/{id}/cancel - Cancel appointment
GET    /api/patient/profile        - Get patient profile
PUT    /api/patient/profile        - Update patient profile
GET    /api/patient/treatments     - Get treatment history
GET    /api/patient/treatments/{id} - Get treatment details
POST   /api/patient/export-treatments      - Export treatment data
GET    /api/patient/jobs/{task_id}/status  - Check export job status
GET    /api/patient/download-export/{filename} - Download exported file
GET    /api/patient/doctors/{id}/booked-slots   - Get booked time slots
```

Drive Link for Presentation Video

[https://drive.google.com/drive/folders/15xD3FNCPTndRWtUi3xtINHrJ4EL8Z74A?usp=sharing]