MAD2 Project Report (Sept25 Term)

Name: Aaradhya Moreshwar Kulkarni
Roll No: 23f1001381
Student Email id: 23f1001381@ds.study.iitm.ac.in

Description

Chikitsa HMS is a comprehensive Hospital Management System that streamlines healthcare operations by enabling patients to book appointments with doctors, doctors to manage their schedules and consultations, and administrators to oversee the entire hospital workflow. The system features role-based access control, automated email reminders, and real-time appointment slot management.

Used LLM for bootstrap css, debugging assistance, and architecture suggestions. (15-25%)

Technologies used

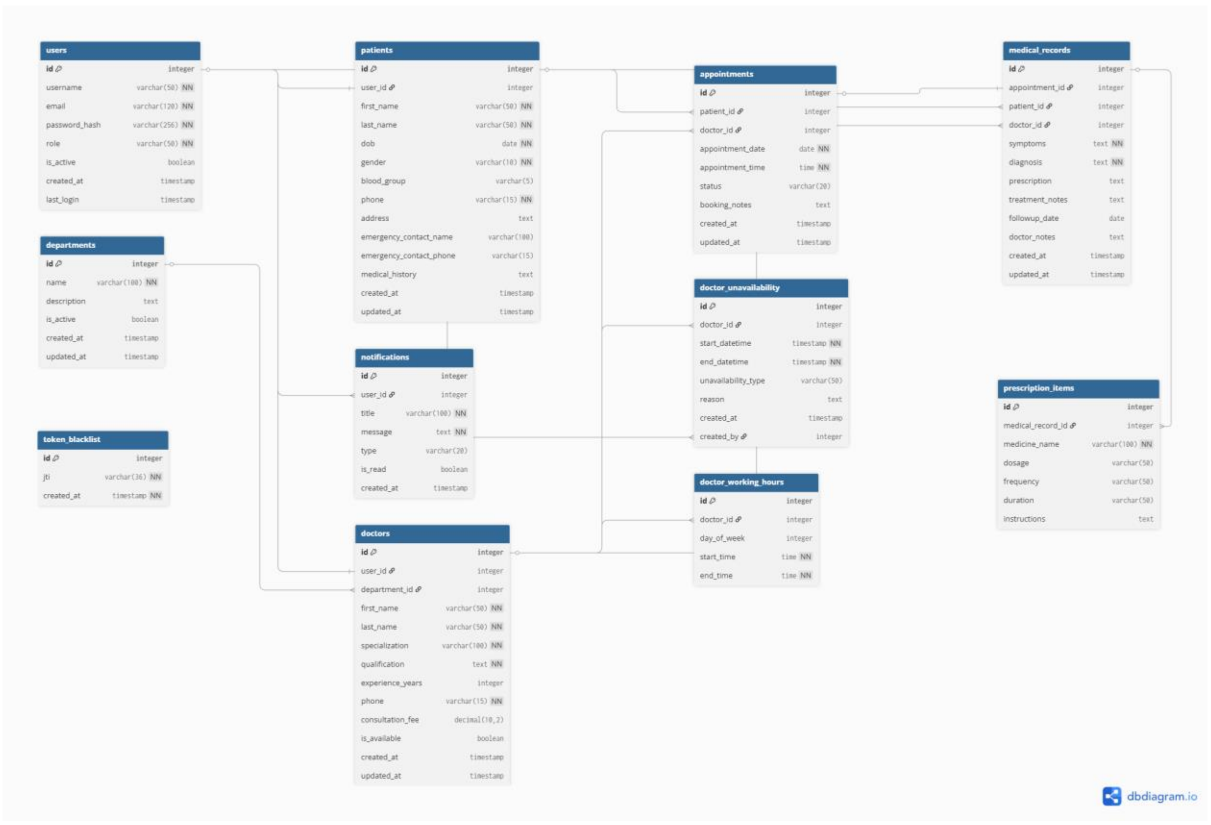
1. Backend: - **Flask:** Core web framework for REST API development

- **Flask-SQLAlchemy:** ORM for database operations and relationship management
- **Flask-JWT-Extended:** JWT-based authentication and authorization
- **Flask-Bcrypt:** Password hashing for secure authentication
- **Flask-Mail:** Email notifications and reminders
- **Flask-CORS:** Cross-origin resource sharing for frontend integration
- **Celery:** Distributed task queue for scheduled jobs (daily reminders, monthly reports)
- **Redis:** Message broker for Celery and response caching
- **SQLite:** Lightweight database for development

2. Frontend: - **Vue.js 3:** Progressive JavaScript framework with Composition API

- **Vue Router:** Client-side routing with navigation guards
- **Vuex:** Centralized state management
- **Bootstrap 5:** Responsive UI framework
- **Bootstrap Icons:** Icon library
- **Axios:** HTTP client for API requests

DB Schema Design



- Normalized schema with proper foreign key relationships
- One-to-One: User ↔ Patient, User ↔ Doctor, Appointment ↔ MedicalRecord
- One-to-Many: Doctor → Appointments, Patient → Appointments, Department → Doctors
- Unique constraints on appointment slots to prevent double-booking
- Soft delete support via is_active boolean fields
- Audit timestamps (created_at, updated_at) on all entities

API Design:

11. Authentication (/auth)

POST /auth/login	: User login with role-based access
POST /auth/register/patient	: Patient self-registration
POST /auth/logout	: Logout and token blacklisting
GET /auth/me	: Get current user info
GET /auth/profile	: Get user profile
PUT /auth/profile	: Update user profile

2. Admin – Dashboard (/admin)

GET /admin/dashboard/stats	: Dashboard statistics

3. Admin – Departments (/admin/departments)

GET /admin/departments	: List all departments
POST /admin/departments	: Create new department
GET /admin/departments/<id>	: Get department by ID
PATCH /admin/departments/<id>	: Update department
DELETE /admin/departments/<id>	: Delete department
GET /admin/departments/<id>/records	: Get department medical records

4. Admin – Doctors (/admin/doctors)

GET /admin/doctors	: List all doctors
POST /admin/doctors	: Create new doctor
GET /admin/doctors/<id>	: Get doctor by ID
PATCH /admin/doctors/<id>	: Update doctor
GET /admin/doctors/<id>/working-hours	: Get doctor working hours
POST /admin/doctors/<id>/working-hours	: Create working hours
PUT /admin/doctors/<id>/working-hours	: Bulk update working hours
PATCH /admin/doctors/<id>/working-hours/<day>	: Update specific day
DELETE /admin/doctors/<id>/working-hours	: Delete all working hours
GET /admin/doctors/<id>/unavailability	: Get doctor unavailability

5. Admin – Patients (/admin/patients)

GET /admin/patients	: List all patients
POST /admin/patients	: Create new patient
GET /admin/patients/<id>	: Get patient by ID
PATCH /admin/patients/<id>	: Update patient
GET /admin/patients/<id>/history	: Get patient medical history
POST /admin/<id>/export-records	: Export patient records via email

6. Admin – Appointments (/admin/appointments)

GET /admin/appointments	: List all appointments
GET /admin/appointments/<id>	: Get appointment by ID

PATCH /admin/appointments/<id>/status : Update appointment status
GET /admin/appointments/<id>/record : Get appointment medical record

7. Admin – Medical Records (/admin/records)
GET /admin/records : List all medical records
GET /admin/records/<id> : Get medical record by ID

8. Admin – User & Hospital Management
PATCH /admin/users/<id>/status : Toggle user active status
DELETE /admin/users/<id> : Delete user
POST /admin/hospital-holiday : Create hospital holiday
DELETE /admin/hospital-holiday : Remove hospital holiday

9. Doctor Portal (/doctor)
GET /doctor/profile : Get own profile
PUT /doctor/profile : Update own profile
GET /doctor/dashboard/stats : Dashboard statistics
GET /doctor/patients : List consulted patients
GET /doctor/patients/stats : Get patients statistics
GET /doctor/patients/<id>/stats : Get specific patient stats
GET /doctor/patients/<id>/records : Get patient records
GET /doctor/patients/<id>/history : Get patient medical history
GET /doctor/appointments : List own appointments
GET /doctor/appointments/<id> : Get appointment details
PATCH /doctor/appointments/<id>/status : Update appointment status
POST /doctor/appointments/<id>/complete : Complete appointment with record
GET /doctor/appointments/<id>/record : Get appointment medical record
POST /doctor/appointments/<id>/record : Create medical record
GET /doctor/records : List own medical records
GET /doctor/records/<id> : Get medical record by ID
PUT /doctor/records/<id> : Update medical record
POST /doctor/records/<id>/prescription : Add prescription item
PUT /doctor/prescription/<id> : Update prescription item
DELETE /doctor/prescription/<id> : Delete prescription item
GET /doctor/working-hours : Get own working hours
GET /doctor/unavailability : List unavailability periods
POST /doctor/unavailability : Create unavailability
PUT /doctor/unavailability/<id> : Update unavailability
DELETE /doctor/unavailability/<id> : Delete unavailability
GET /doctor/calendar : Get calendar view
GET /doctor/schedule/<date> : Get daily schedule

10. Patient Portal (/patient)

GET /patient/profile : Get own profile
PUT /patient/profile : Update own profile
GET /patient/appointments : List own appointments
POST /patient/appointments : Book new appointment
GET /patient/appointments/<id> : Get appointment details
PUT /patient/appointments/<id>/reschedule : Reschedule appointment
PUT /patient/appointments/<id>/notes : Update booking notes
POST /patient/appointments/<id>/cancel : Cancel appointment
GET /patient/appointments/<id>/record : Get appointment medical record
GET /patient/records : List own medical records
GET /patient/records/<id> : Get medical record details
POST /patient/export-records : Export records via email

11. Appointment Slots (/appointments)
GET /appointments/slots/<doctor_id> : Get available slots (requires ?date=YYYY-MM-DD)

Architecture and Features

```
chikitsa/
├── backend/
│   ├── core/                # Shared components
│   │   ├── models.py        # SQLAlchemy models (10 tables)
│   │   ├── config.py        # Application configuration
│   │   ├── database.py      # Database initialization
│   │   ├── auth.py          # Role-based decorators
│   │   ├── mail.py          # Email service
│   │   ├── cache.py         # Redis caching utility
│   │   ├── logger.py        # Logging configuration
│   │   └── celery_config.py # Celery task scheduler
│   │
│   ├── auth/                # Authentication module
│   │   ├── routes.py        # Auth endpoints
│   │   └── schema.py        # Pydantic schemas
│   │
│   ├── services/            # Business logic modules
│   │   ├── admin/           # Admin CRUD operations
│   │   ├── doctors/         # Doctor portal services
│   │   ├── patients/        # Patient portal services
│   │   ├── appointments/    # Appointment management
│   │   └── medical_records/ # Medical records handling
│   │
│   ├── utils/               # Utilities
│   │   ├── tasks.py         # Celery background tasks
│   │   ├── driver.py        # Email reminder logic
│   │   ├── email_templates.py  # HTML email templates
│   │   └── report_templates.py # Report generation
│   │
│   ├── app.py               # Flask application factory
│   └── celery_worker.py     # Celery worker entry point
│
├── frontend/
│   ├── src/
│   │   ├── views/
│   │   │   ├── admin/       # Admin dashboard views
│   │   │   ├── doctor/      # Doctor portal views
│   │   │   ├── patient/     # Patient portal views
│   │   │   └── auth/        # Login/Register views
│   │   │
│   │   ├── components/      # Reusable components
│   │   │   ├── ui/          # Generic UI components
│   │   │   ├── common/      # Shared components
│   │   │   ├── admin/         # Admin-specific components
│   │   │   ├── doctor/        # Doctor-specific components
│   │   │   └── patient/       # Patient-specific components
│   │   │
│   │   ├── services/          # API service layer
│   │   ├── store/             # Vuex state management
│   │   └── router/            # Vue Router configuration
│   │
│   └── public/                # Static assets
│
├── api_specification.yaml     # OpenAPI 3.0 specification
└── README.md
```

Implemented Features

1. Core Features

- JWT-based authentication with token blacklisting
- Role-based access control (Admin, Doctor, Patient)
- Real-time appointment slot availability
- Automatic conflict detection for bookings
- Email notifications for appointments

2. Admin Features

- Dashboard with comprehensive statistics
- Department CRUD with doctor count tracking
- Doctor management with working hours configuration
- Patient registration and management
- Appointment monitoring and status updates
- Medical records oversight

3. Doctor Features

- Personal dashboard with today's appointments
- Patient list with consultation history
- Appointment management (complete, cancel, reschedule)
- Medical record creation with prescriptions
- Working hours and unavailability management
- Calendar view for schedule overview

4. Patient Features

- Self-registration with profile management
- Doctor browsing by department
- Real-time slot checking and booking
- Appointment rescheduling and cancellation
- Medical records access with export capability
- Appointment reminders via email

5. Background Jobs (Celery + Redis)

- Daily appointment reminders (7 AM)
- Monthly doctor activity reports (1st of month)

- Configurable schedule via environment variables

6. Performance Optimization

- Redis caching for frequently accessed endpoints
- Cache invalidation on data mutations

Video

https://drive.google.com/file/d/1XdESMyjUN9TFxUHCAzOGJS30EvLZmNVj/view?usp=sharing