Project Documentation

DocSpot — Seamless Appointment Booking for Health

1. Introduction

Project Title: DocSpot — Seamless Appointment Booking for Health
Team ID: LTVIP2026TMIDS88461
Team Size: 3
Team Leader: Jaya Prakash Maneri
Team Member: Katta Mounika
Team Member: Srinivasulu Sake

2. Project Overview

DocSpot is a production-ready MERN application for online doctor appointment booking. Patients can browse doctors with advanced filtering, book appointments with real-time slot validation, and manage their bookings. Doctors apply for approval and manage their schedules. Admins oversee the platform with analytics dashboards.

Key Features:

- Appointment Slot Management: Prevents double bookings, validates against doctor's working hours
- Doctor Availability & Working Hours: Doctors set available days and time ranges; appointments validate against this
- Proper Appointment Status Flow: Pending → Approved → Scheduled → Completed with timestamps
- Doctor Profile Page: Full bio, specialization, experience, fees, and ratings
- Service Layer Pattern: Clean separation of routes, controllers, services, models
- Centralized Error Handling: Custom AppError class with proper HTTP status codes
- Role-Based Access Control: Separate middleware for Admin, Doctor, User
- JWT Authentication: Secure endpoints with token-based auth
- Search & Filtering: Find doctors by name, specialization, fees
- Pagination: Efficient data loading
- Appointment Reschedule: Users can reschedule

- Admin Analytics Dashboard: Real-time stats

3. Architecture

Frontend:

- Built with React 18, React Router, Axios, Bootstrap 5, Ant Design
- Component-based structure
- Routing via AppRouter.jsx
- State management using Context API
- Responsive UI for all devices

Backend:

- Node.js and Express.js
- RESTful API endpoints
- Middleware for authentication, error handling, and role-based access control
- Modular controllers and services
- Centralized error handler

Database:

- MongoDB with Mongoose ODM
- Schemas: User, Doctor, Appointment
- Proper schema design with references and audit trail

4. Setup Instructions

Prerequisites:

- Node.js
- MongoDB

Installation Steps:

1. Clone the repository
2. Install dependencies in frontend and backend folders using npm install

3. Set up environment variables as described in README.md
4. Start MongoDB locally or connect to a cloud instance

Quick Start:

Quick Start:

- Install all dependencies:

  npm run install:all

- Configure environment:

  cd backend

  cp .env.example .env

  # Edit .env with your MongoDB URI and JWT secret

- Seed demo data:

  npm run backend:seed

- Run both servers:

  npm run dev:all

Frontend (Vercel URL) : https://medi-connect-seamless-appointment-b.vercel.app/

Backend: https://docspot-seamless-appointment-booking-forw39m.onrender.com/api

5. Folder Structure

Client (frontend/):

- src/components/: UI components for admin, doctor, user, common
- src/pages/: Page views
- src/services/: API calls
- src/styles/: CSS files
- src/utils/: Utility functions

Server (backend/):

- config/: Database connection

- controllers/: Request handlers
- middlewares/: Auth, roles, error handling
- models/: Mongoose schemas
- routes/: API endpoints
- services/: Business logic
- utils/: Custom errors
- uploads/: User documents
- index.js: Express server

6. Running the Application

Frontend:

```
cd frontend
npm start
```

Backend:

```
cd backend
npm start
```

7. API Documentation

Users:

- POST /api/users/register — Register new user
- POST /api/users/login — Login user

Doctors:

- GET /api/doctors — List approved doctors (paginated, searchable)
- GET /api/doctors/:id — Get doctor full profile
- GET /api/doctors/availability/check — Check if slot is available
- POST /api/doctors/apply — Apply as doctor
- GET /api/doctors/:id/appointments — Get doctor's appointments

Appointments:

- POST /api/appointments/book — Book appointment (with slot validation)
- GET /api/appointments/me — Get user's appointments (paginated)
- GET /api/appointments/all — Get all appointments (admin only, filterable)
- PUT /api/appointments/status/:id — Update status (admin/doctor only)
- PUT /api/appointments/reschedule/:id — Reschedule appointment

Admin:

- GET /api/admin/stats — Get platform statistics
- GET /api/admin/pending-doctors — List pending approvals
- POST /api/admin/approve-doctor/:id — Approve doctor
- POST /api/admin/reject-doctor/:id — Reject doctor

8. Authentication

- JWT-based authentication
- Middleware for route protection
- Token stored in client for session management
- Role-based access control for admin, doctor, user

9. User Interface

- Modern, responsive UI
- Users can register, log in, and book appointments
- Doctors manage schedules and credentials
- Admins access dashboards for oversight
- Screenshots and GIFs available in project documentation

10. Testing

- Manual and automated testing
- Backend scripts in backend/scripts/

- Unit, integration, and end-to-end tests using Jest and Mocha

11. Screenshots or Demo

- demo link in https://drive.google.com/file/d/1l2GIAVBAJUJ6TdQWgtPZA_cZx-LefGPZ/view?usp=drive_link
- Visuals showcase booking flow, admin dashboard, doctor management

12. Known Issues

Common Issues

- Duplicate email registration is not allowed; users see "Email already registered."
- Weak passwords or missing fields during registration prompt validation errors.
- Login with incorrect credentials returns "Invalid credentials" or "User not found."
- Doctor applications may remain pending if not approved by admin.
- Appointment booking may fail if the slot is already taken or outside doctor's working hours.
- Admin dashboard statistics may not update in real-time if backend is not running.

Troubleshooting Steps

1. **Login Issues**
   - Restart the backend server:
     - Windows: Double-click START_SERVER.bat
     - Mac/Linux: Run npm start in backend directory
   - Restart MongoDB service.
   - Refresh your browser (Ctrl+F5 or Cmd+Shift+R).
   - Clear browser cache and session storage.
2. **Backend Not Responding**
   - Visit http://localhost:5000/api/health in your browser.
   - If you see a JSON status, backend is running.
   - If connection is refused, restart the backend.

3. **Database Connection Errors**
    - Ensure MongoDB is running locally or your connection string is correct.
    - Check .env file for correct MongoDB URI.
4. **Frontend Not Loading**
    - Make sure you have run npm install in the frontend directory.
    - Start the frontend with npm start in the frontend directory.
5. **Appointment Booking Fails**
    - Ensure the doctor is approved and has available slots.
    - Check for overlapping appointments.

13. Future Enhancements

- Video consultation integration
- Advanced analytics dashboard
- Mobile app version for iOS and Android