Doc Spot – Full Stack MERN Project Documentation

1. Introduction

Project Title: DocSpot – Online Doctor Appointment Booking System

Team Member: Vignesh

2. Project Overview

Purpose:

DocSpot is designed to digitize and simplify the entire doctor appointment booking process. The primary purpose of the system is to eliminate traditional manual booking methods by offering a secure, fast, and user-friendly online platform. It provides patients, doctors, and administrators with seamless access to essential features such as real-time appointment scheduling, profile management, and data tracking.

DocSpot aims to:

- Allow patients to easily find doctors based on specialization, availability, and location.
- Provide a structured platform for doctors to manage appointments and patient interactions.
- Enable administrators to manage users, doctors, approvals, and overall platform activities.
- Improve healthcare workflow efficiency by reducing delays and manual effort.
- Ensure secure and authenticated interactions using a robust MERN-based infrastructure.

Features:

- User authentication (JWT-based)
- Patient dashboard and appointment booking
- Doctor dashboard for managing schedules and appointments
- Admin panel to manage doctors, users, and appointments
- Real-time notifications
- Secure API integration

3. Architecture

Frontend (React.js):

- React with Hooks and Context API
- Axios for API calls
- Bootstrap/Material UI for styling
- Protected routes using React Router

Backend (Node.js + Express.js):

- RESTful API
- Authentication middleware
- Role-based access control

Database (MongoDB + Mongoose):

- Collections: Users, Doctors, Appointments
- Schema validation using Mongoose models

4. Setup Instructions

Prerequisites:

- Node.js (v16+)
- MongoDB or MongoDB Atlas
- Git

Installation Steps:

1. Clone the repository.

2. Install dependencies for frontend & backend: npm install

3. Configure .env in backend:

- MONGO_DB="mongodb_connection_string"
- JWT_SECRET="your_secret"

4. Start frontend & backend.

5. Folder Structure

- **Client (React):**
  - /src/components – UI Components
  - /src/pages – Screens
  - /src/context – Global state
  - /src/utils – Helper functions

- **Server (Node.js):**
  - /routes – API routes
  - /controllers – Business logic
  - /models – Mongoose schemas
  - /middleware – Auth & security layers

6. Running the Application

Frontend: npm start

Backend: nodemon index.js

7. API Documentation

- **User APIs:**

- POST /api/user/register – Register user
- POST /api/user/login – Login user
- GET /api/user/profile – Get user profile

- **Doctor APIs:**

- POST /api/doctor/apply – Apply as doctor
- GET /api/doctor/appointments – Get doctor appointments
- POST /api/doctor/update-status – Update appointment status

- **Appointment APIs:**

- POST /api/appointment/book – Book appointment
- GET /api/appointment/user-appointments – User appointments
- POST /api/appointment/check – Check availability

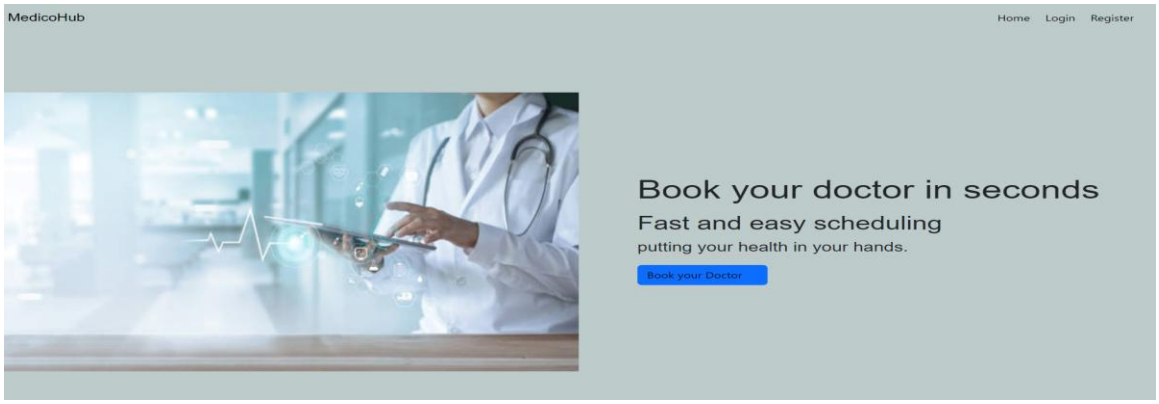
➢ **Admin APIs**

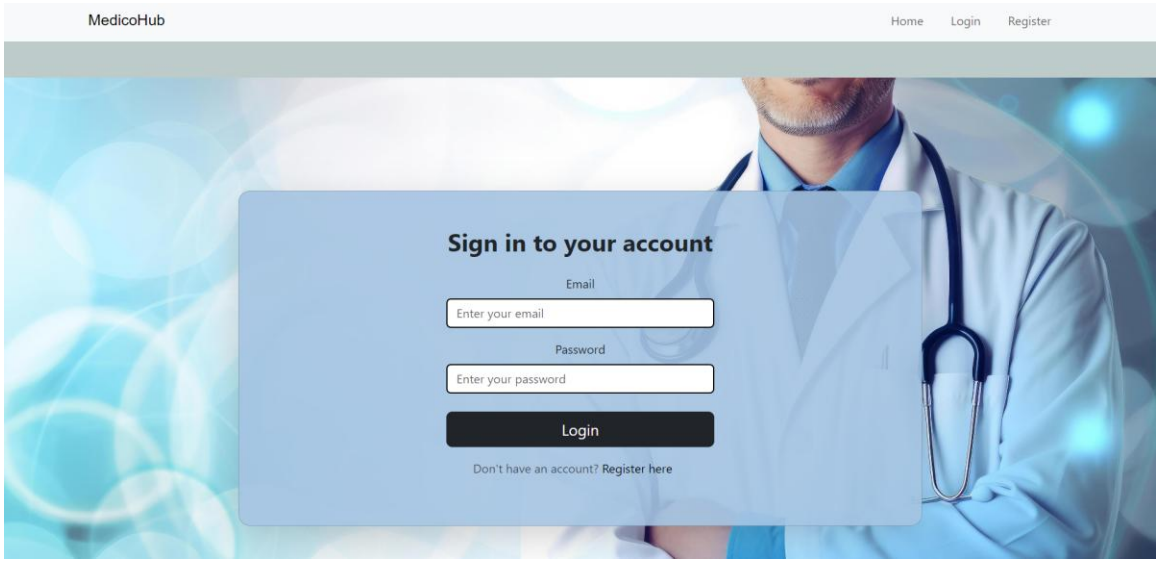
- GET /api/admin/get-all-doctors – Fetch all doctors
- POST /api/admin/update-doctor-status – Approve/Reject doctor
- GET /api/admin/get-all-users – Fetch all users
- GET /api/admin/all-appointments – Fetch all appointments

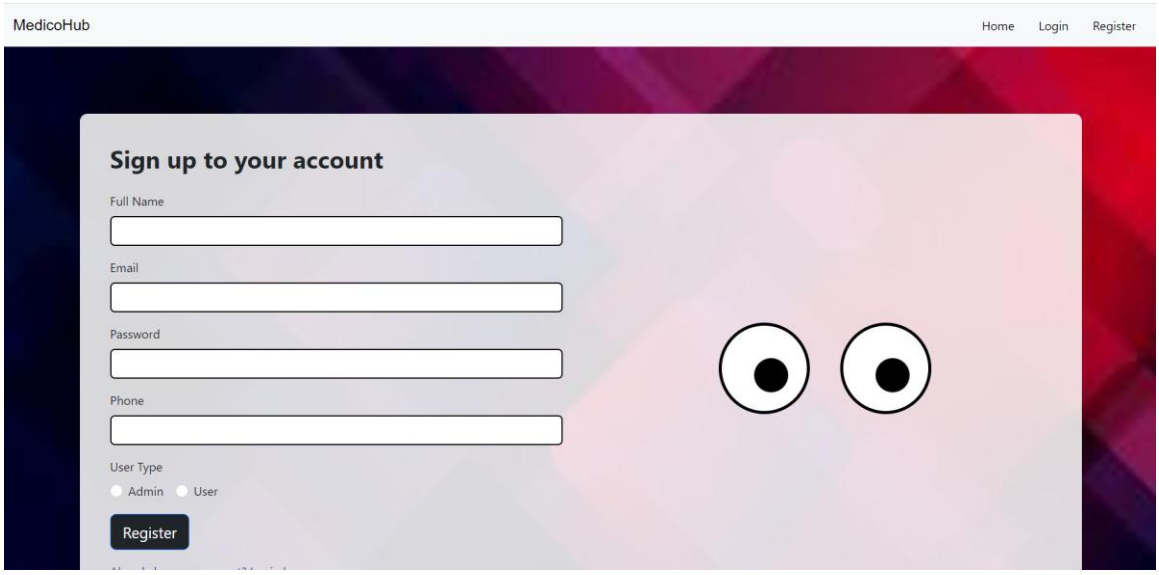
8. Authentication:

- JWT-based authentication
- Tokens stored in local Storage
- Role validation: user, doctor, admin

9. User Interface:







10. Testing:

- Manual testing
- JWT verification and API endpoint tests

11. Screenshots or Demo:

(Add demo/video links)

12. Known Issues:

- Occasional delay in backend responses
- Admin panel UI improvements needed

13. Future Enhancements:

- Add payment integration
- Add video consultation feature
- Implement analytics dashboards