**Full Stack Development with MERN**

**Book a Doctor**

**DOCUMENTATION**

**DocSpot: Seamless Appointment Booking for Health**

**1. Introduction**

**Project Title:** DocSpot: Seamless Appointment Booking for Health  
**Team Members:** udumula sai Sudheer reddy(238X1A45C2) , Velaga sri Varshini(238X1A05F5), Sanka Giri Siva Santosh Gupta , Talari Bhanu Prasad

| **Name** | **Role** |
| --- | --- |
| Alice Smith | Frontend Developer |
| Bob Johnson | Backend Developer |
| Carol Lee | UI/UX Designer |
| David Kim | Database Engineer |
| Emma Brown | Project Manager |

**2. Project Overview**

**Purpose**

DocSpot is a web-based platform that simplifies the process of booking medical appointments by connecting patients with healthcare providers in a user-friendly interface. It enables real-time scheduling, secure user authentication, and streamlined management of appointments.

**Features**

* User authentication (patients and doctors)
* Browse/search doctors by specialty, location, availability
* Book, reschedule, or cancel appointments
* Doctor dashboard to manage schedules and view bookings
* Email notifications and reminders
* Responsive, mobile-friendly design

**3. Architecture**

**Frontend (React)**

* Built using React.js with functional components and hooks
* React Router for navigation
* Axios for API communication
* Redux for global state management (optional)

**Backend (Node.js & Express.js)**

* RESTful API developed with Express.js
* JWT-based authentication middleware
* Controllers for managing appointments, users, doctors
* Centralized error handling

**Database (MongoDB)**

* MongoDB Atlas used as a cloud-hosted NoSQL database
* Mongoose ODM to define schema and interact with data

**Schemas:**

* **User**: name, email, password, role (doctor/patient), etc.
* **Doctor**: user reference, specialty, availability
* **Appointment**: patient ID, doctor ID, date, time, status

**4. Setup Instructions**

**Prerequisites**

* Node.js (v16+)
* npm
* MongoDB (local or MongoDB Atlas)

**Installation**

bash

CopyEdit

# Clone the repository

git clone https://github.com/your-repo/docspot.git

cd docspot

# Setup server

cd server

npm install

# Add .env file with DB\_URL, JWT\_SECRET, etc.

# Setup client

cd ../client

npm install

# Add .env file with REACT\_APP\_API\_URL

**5. Folder Structure**

**Client (React)**

pgsql

CopyEdit

client/

│

├── public/

├── src/

│ ├── components/ # Reusable components (Header, Footer, etc.)

│ ├── pages/ # Pages (Home, Login, Profile, Booking)

│ ├── services/ # API calls using Axios

│ ├── context/ # Auth context or global state

│ ├── App.js

│ └── index.js

**Server (Node.js)**

bash

CopyEdit

server/

│

├── controllers/ # Logic for routes

├── models/ # Mongoose schemas

├── routes/ # Route definitions

├── middleware/ # Auth and error handlers

├── config/ # DB connection

├── utils/ # Helper functions

├── .env

├── server.js

**6. Running the Application**

**Frontend**

bash

CopyEdit

cd client

npm start

**Backend**

bash

CopyEdit

cd server

npm start

**7. API Documentation**

| **Endpoint** | **Method** | **Description** | **Auth Required** | **Request Body / Params** | **Sample Response** |
| --- | --- | --- | --- | --- | --- |
| /api/auth/register | POST | Register a new user | No | { name, email, password, role } | 201 Created |
| /api/auth/login | POST | Login and receive JWT token | No | { email, password } | { token, user } |
| /api/doctors | GET | Get list of doctors | No | N/A | [ { id, name, specialty } ] |
| /api/appointments | POST | Book appointment | Yes | { doctorId, date, time } | 201 Created |
| /api/appointments/me | GET | Get logged-in user's appointments | Yes | Token in header | [ { id, status, date } ] |

**8. Authentication**

* Uses **JWT** (JSON Web Tokens) stored in localStorage.
* Tokens are required for protected routes and verified via middleware.
* Passwords are hashed using bcrypt.
* Role-based access control for patient and doctor.

**9. User Interface**

**Patient Dashboard**

* View upcoming appointments
* Book or cancel appointments

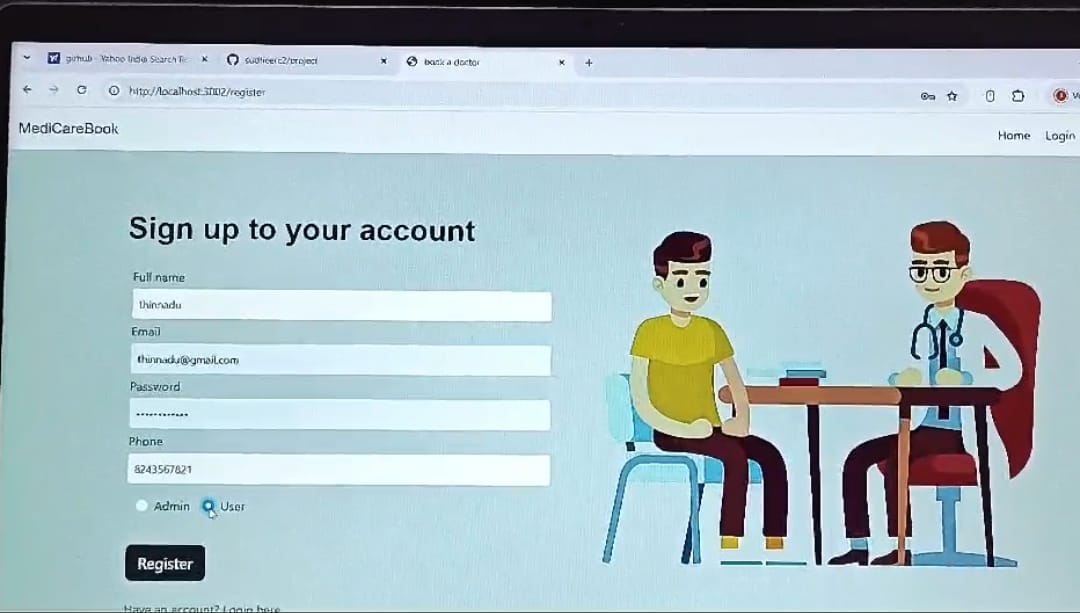
**Doctor Dashboard**

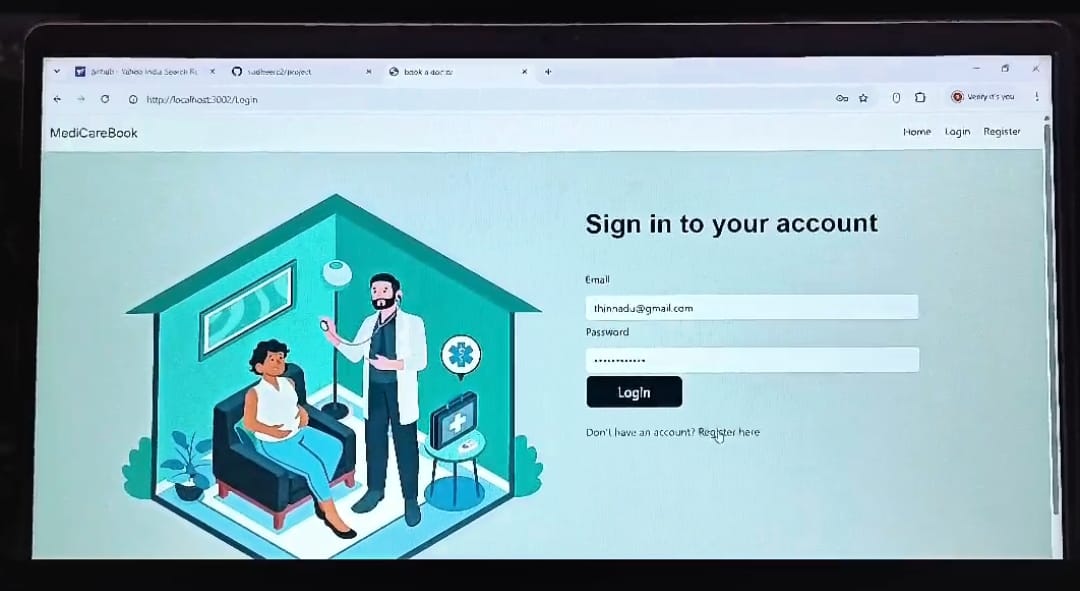
* View scheduled appointments
* Set or edit availability

**10. Testing**

* Unit Testing: **Jest** for backend APIs
* Frontend Testing: **React Testing Library**
* Postman used for manual API testing

**11. Screenshots or Demo**:





**12. Known Issues**

* Appointment conflicts not checked in real-time in some edge cases
* Limited mobile support on low-resolution devices
* Email notifications may delay occasionally (SMTP limits)

**13. Future Enhancements**

* Integrate real-time chat between patient and doctor
* Video consultations via WebRTC
* Push notifications for mobile users
* Admin panel for monitoring system usage
* Improved calendar UI with drag & drop rescheduling