**DocSpot**:

🩺 **Full Stack Development with MERN**

📄 **Project Documentation - DocSpot**

**1. Introduction**

* **Project Title**: *DocSpot - Seamless Appointment Booking for Health*
* **Team Members**:
  + Anushya Kadali
  + Ashoksrinivassivakiran Gangalakurthi
  + Hema Satya Nikhil Ainavilli

**2. Project Overview**

* **Purpose**:   
   DocSpot simplifies the process of booking appointments with healthcare professionals. It enables users to find doctors, schedule appointments, and conduct teleconsultations online.
* **Features**:
  + Doctor and Patient Login
  + Real-Time Appointment Booking
  + Teleconsultation (Video)
  + Admin Dashboard
  + Notifications & Reminders
  + Doctor Availability Management

**3. Architecture**

* **Frontend**:   
   Built with React.js, React Router for navigation, Axios for API requests, and Tailwind CSS for styling.
* **Backend**:   
   Node.js with Express.js for REST APIs. JSON Web Tokens (JWT) for authentication and authorization.
* **Database**:   
   MongoDB is used for storing user data, appointment schedules, and doctor profiles. Mongoose is used for schema design and queries.

**4. Setup Instructions**

* **Prerequisites**:
  + Node.js >= 14
  + MongoDB
  + Git
* **Installation**:
* git clone https://github.com/Nikhil-193/DOCSPOT-DOCTOR\_APPOINTMENT-NIKHIL.git
* cd DOCSPOT-DOCTOR\_APPOINTMENT-NIKHIL
* cd client
* npm install
* cd ../server
* npm install
* Create a .env file in the server directory with:
* PORT=5000
* MONGO\_URI=your\_mongodb\_connection\_string
* JWT\_SECRET=your\_jwt\_secret

**5. Folder Structure**

* **Client** (React):
* /client
* ├── public/
* ├── src/
* │ ├── components/
* │ ├── pages/
* │ ├── context/
* │ ├── App.js
* │ └── index.js
* **Server** (Node.js):
* /server
* ├── controllers/
* ├── models/
* ├── routes/
* ├── middleware/
* ├── config/
* ├── server.js

**6. Running the Application**

# Frontend

cd client

npm start

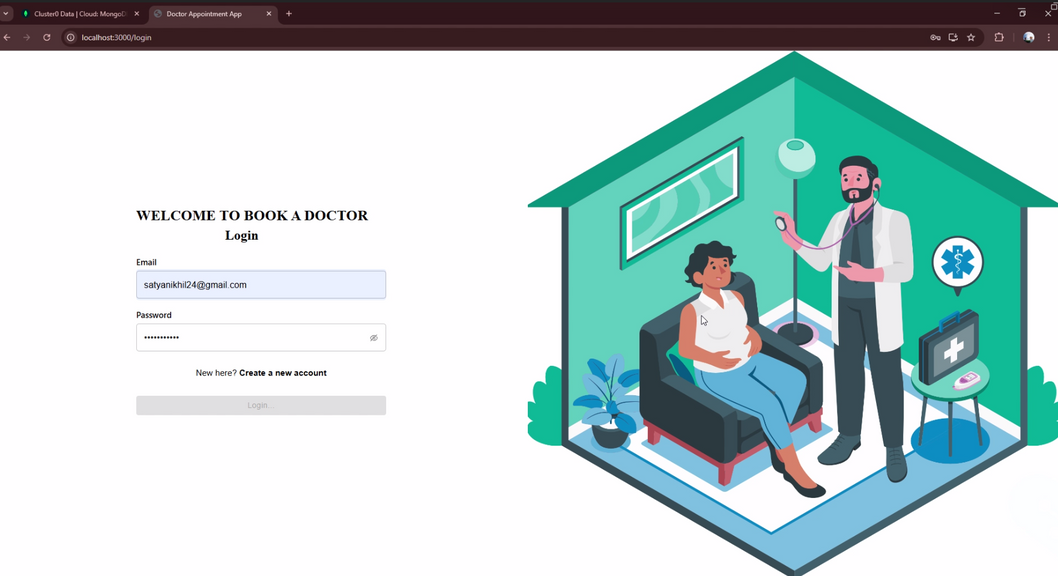
# Backend

cd server

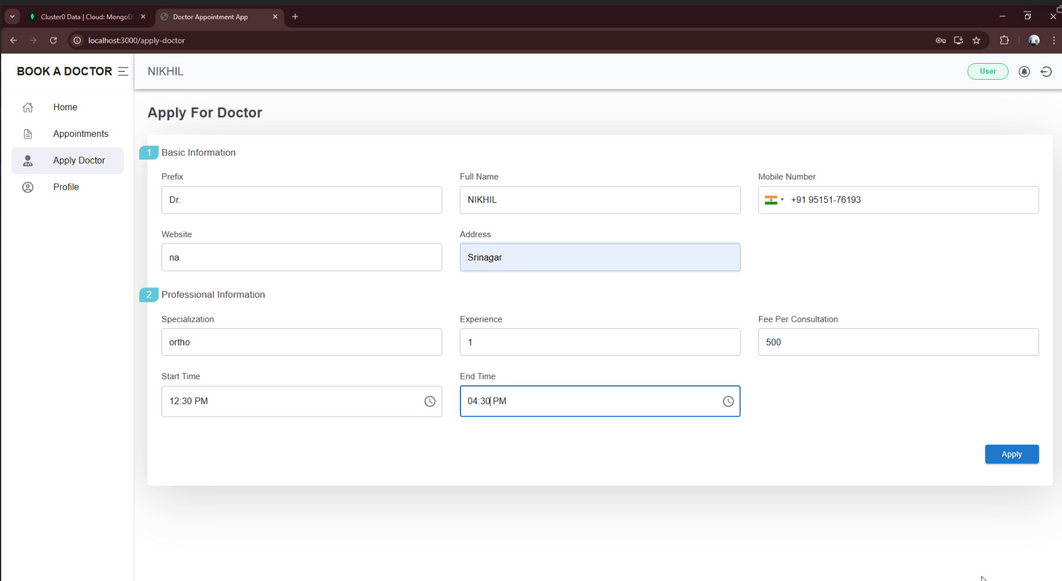
npm start

**7. API Documentation**

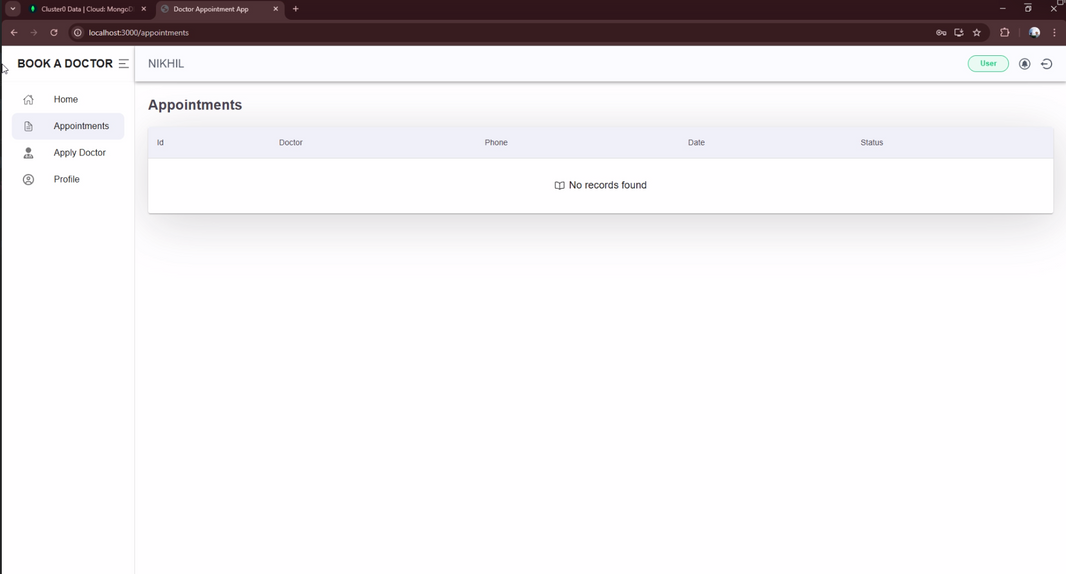
* **POST /api/users/login** – User login



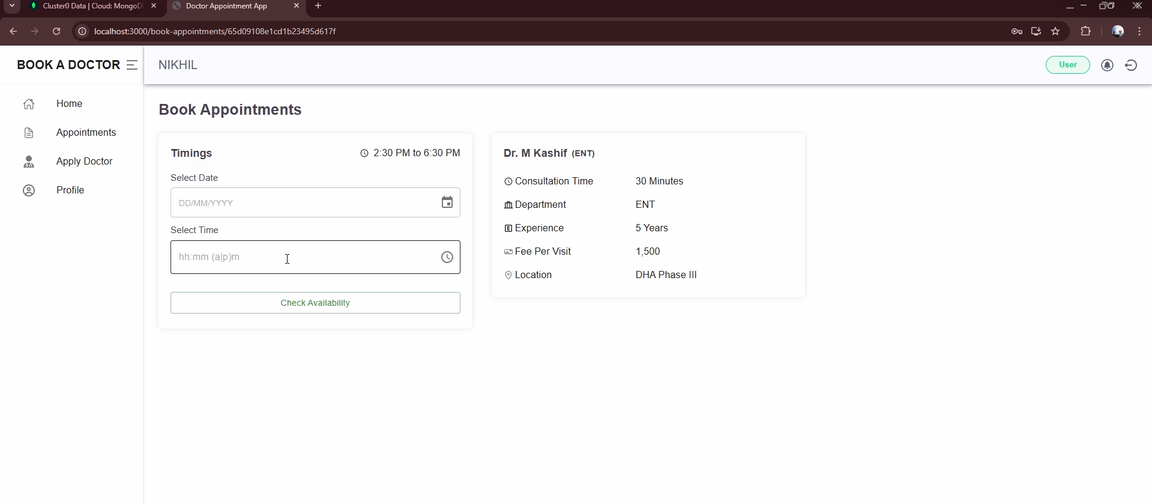
* **POST /api/doctors/add** – Add new doctor



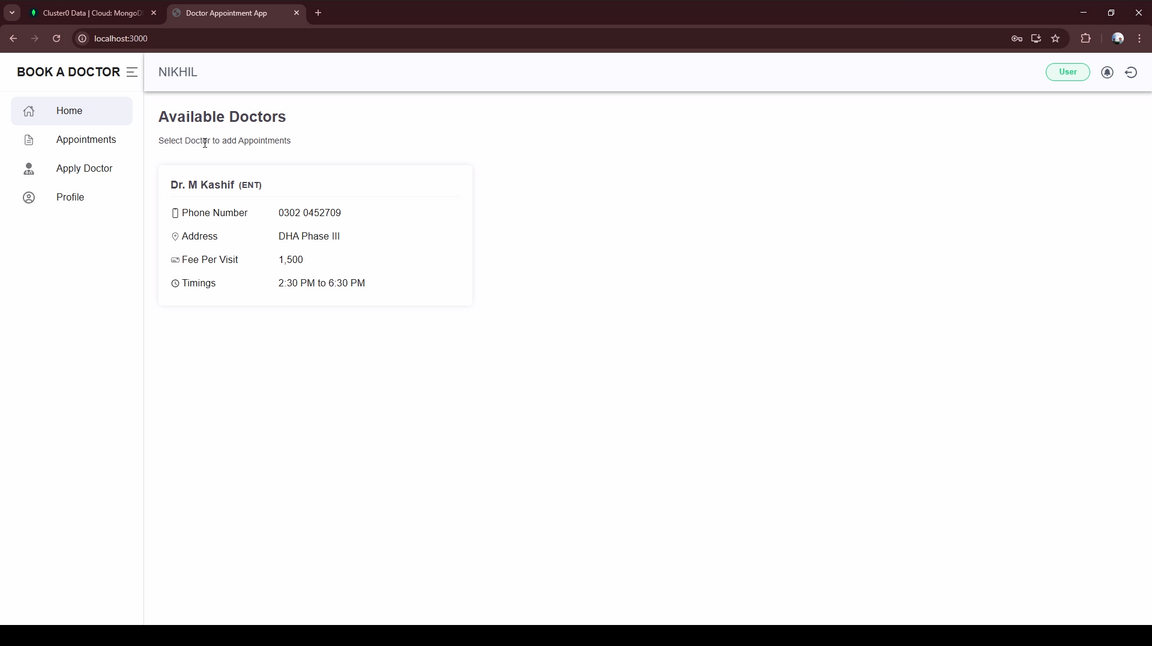
* **GET /api/appointments/:userId** – Fetch user appointments



* **POST /api/appointments/book** – Book an appointment



* **GET /api/doctors/list** – List available doctors

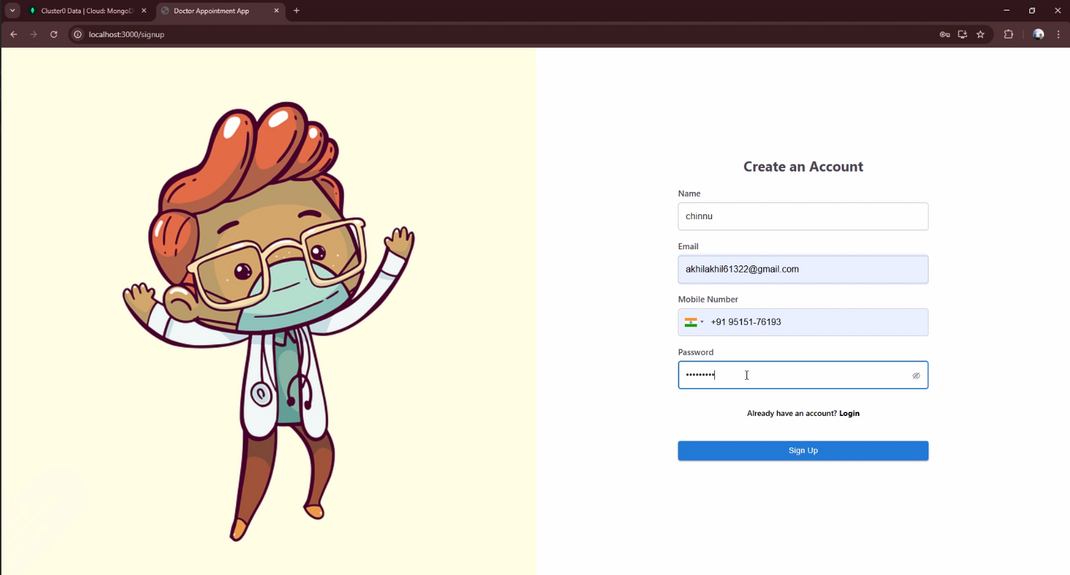


**8. Authentication**

* Handled using JWT.
* On login, users receive a token stored in localStorage.
* Protected routes validated using middleware on the server.

**9. User Interface**

* Simple and intuitive interface with:
  + Doctor search page



* + Booking calendar
  + Dashboard for users and doctors
  + Admin panel for managing users/doctors

(Screenshots can be added here or links to demo video.)

**10. Testing**

* **Tools Used**:
  + Postman (API testing)
  + Manual testing for UI
  + Jest for unit testing (planned)
* **Strategy**:
  + Unit tests for backend
  + End-to-end tests for major flows

**11. Screenshots or Demo**

* 🔗 [Live Demo](https://drive.google.com/file/d/1PC9rXaphVxfmt9fXmiBIXqEBvMmQbyjz/view?usp=drive_link)  : [Cluster0 Data \_ Cloud\_ MongoDB Cloud - Google Chrome 2025-06-26 18-57-49.mp4](https://drive.google.com/file/d/1PC9rXaphVxfmt9fXmiBIXqEBvMmQbyjz/view?usp=drive_link)
* 🖼️ Screenshots: UPLOADED IN GIT HUB LINK

**12. Known Issues**

* Teleconsultation may fail on weak networks.
* Admin panel needs role-based access restrictions.

**13. Future Enhancements**

* Push Notifications
* SMS/Email Appointment Reminders
* Payment Gateway Integration
* Multi-language Support
* Prescription Uploads