**Full Stack Development with MERN**

**Project Documentation: DocSpot**

---

## 1. Introduction

- **Project Title:** DocSpot-Doctor Appointment Booking
- **Team Members:**

  Team ID: LTVIP2025TMID56056

  Team Size: 4

  Team Leader : Karanam Sai Bhavana

  Team member: Chinnadandu Keerthana

  Team member: Shaik Bashira Begum

  Team member: V Reddy Swaroopa

---

## 2. Project Overview

- **Purpose:**
  DocSpot is a web application designed to simplify the process of booking doctor appointments. It allows users to search for doctors based on specialization, view available time slots, and book appointments online. The system also provides doctors with a dashboard to manage appointments and patient details.
- **Features:**
  - User registration and login
  - Doctor profiles and availability
  - Appointment scheduling and management
  - Email confirmation for bookings
  - Admin panel for managing users and doctors

---

## 3. Architecture

- **Frontend:**
  Built with React.js using functional components and React Router for navigation. Redux is used for state management, and Axios handles HTTP requests.

- **Backend:**
  Node.js and Express.js power the REST API. The backend handles authentication, appointment logic, and interactions with the database.
- **Database:**
  MongoDB stores user data, doctor profiles, and appointment information. Mongoose is used for schema design and querying.

---

**4. Setup Instructions**

- **Prerequisites:**
  - Node.js (v16+)
  - MongoDB (local or Atlas cloud instance)
  - npm or yarn package manager
- **Installation:**

```
cd docspot

cd client

npm install

cd ../server

npm install
```

---

**5. Folder Structure**

- **Client:**
  - /src – Contains all React components
  - /pages – Different page views (Home, Login, Book Appointment)
  - /redux – Redux actions and reducers
- **Server:**
  - /controllers – Handles request logic
  - /routes – Defines API routes
  - /models – Mongoose schemas
  - /middleware – Auth and error handling

- /config – DB and environment config

---

**6. Running the Application**

To start the development servers:

- **Frontend:**

cd client

npm start

- **Backend:**

cd server

npm start

---

**7. API Documentation**

**Base URL:** /api

- **POST /auth/register** – Register new users
- **POST /auth/login** – User login
- **GET /doctors** – List all doctors
- **GET /doctors/:id** – Get specific doctor details
- **POST /appointments** – Book an appointment
- **GET /appointments/:userId** – Get user's appointments

Include headers (e.g., Authorization: Bearer <token>) where necessary.

---

**8. Authentication**

- **JWT Tokens** are used for authentication.
- On login, users receive a token to be included in subsequent API requests.
- Role-based access is implemented (User, Doctor, Admin).

---

**9. User Interface**

- Responsive layout using Bootstrap and custom CSS.

- Pages:
    - Home Page
    - Doctor Listings
    - Appointment Form
    - User Dashboard
    - Admin Panel

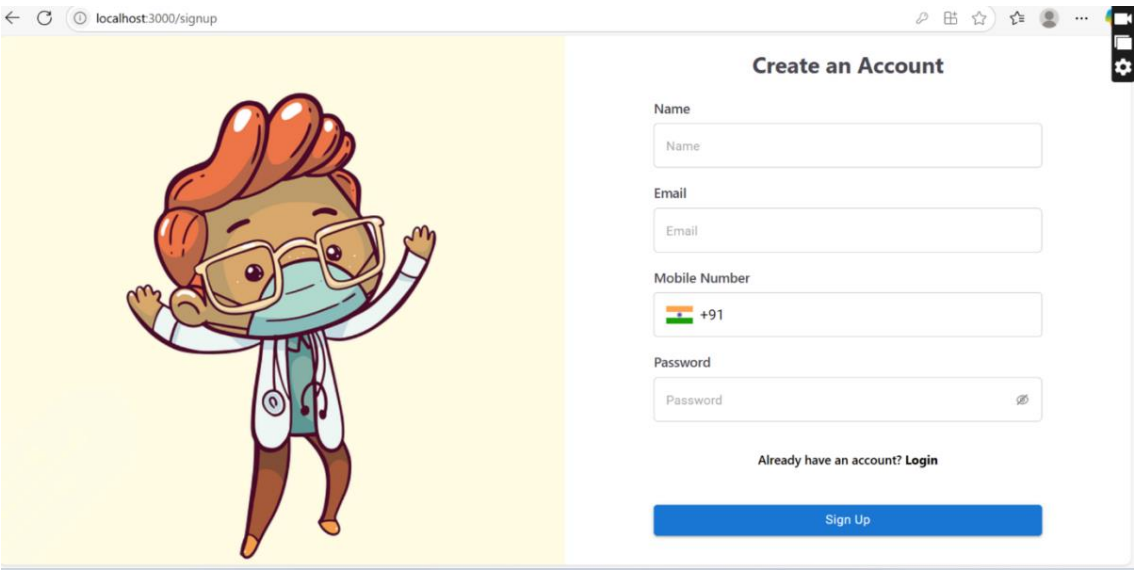
---

**10. Testing**

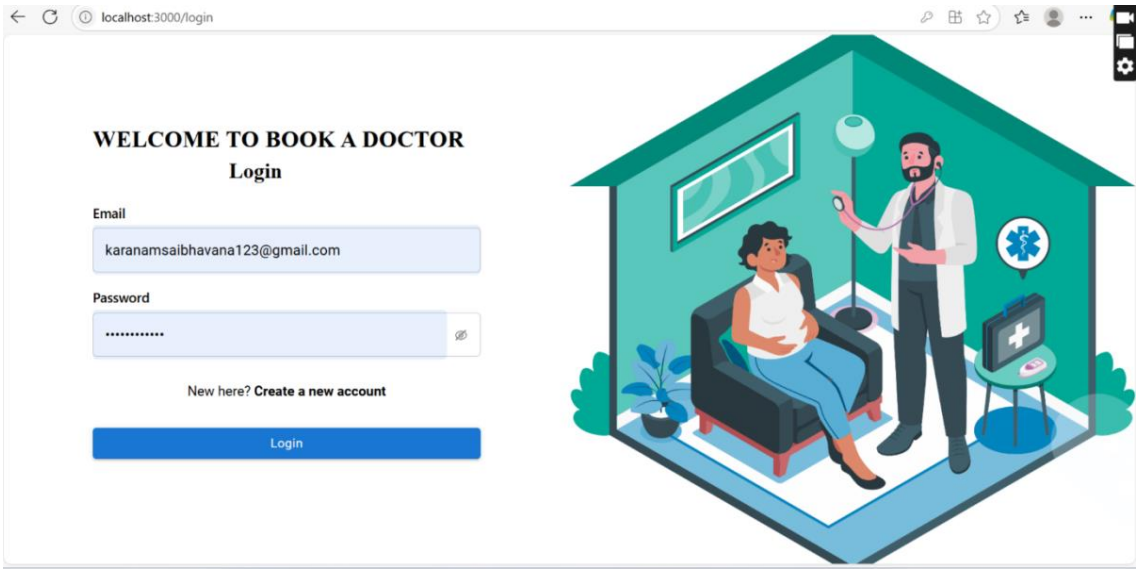
- Unit Testing: Jest
- Integration Testing: Supertest
- Manual testing performed for appointment workflows and user login/logout.

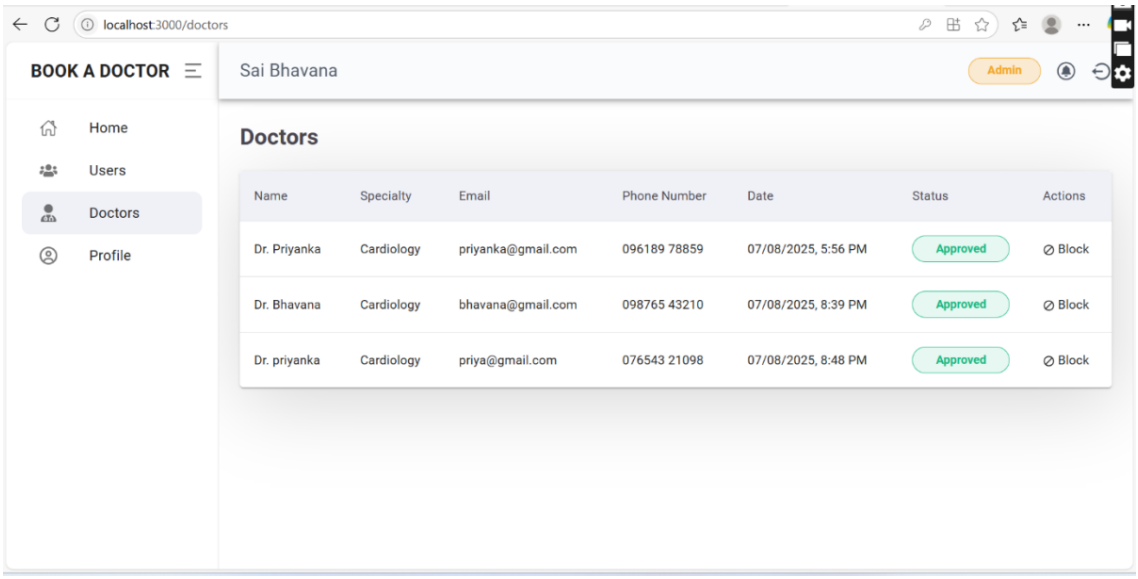
---

**11. Screenshots or Demo**

- Screenshots:
    - Sign-up page
- 

    - Login

- 

- Admin


- ---

## 12. Known Issues

- Calendar UI does not support drag-to-select for time slots
- No password reset functionality yet
- Performance lags slightly with large user data in dashboard view

---

## 13. Future Enhancements

- Implement video consultation via WedRTC

- Enable SMS reminders for upcoming appointments
- Add review and rating system for doctors

---