**1. Introduction**

* **Project Title:** DocSpot – Seamless Appointment Booking for Health
* **Team Members:**
  + *Mitta Sushma* – Team Leader
  + *Kakarla Sudharani-Team member*
  + *Dharma Teja- Team member*

**2. Project Overview**

* **Purpose:**  
  DocSpot is a role-based full-stack web application developed using the MERN stack, aimed at solving the hassle of manual appointment booking by providing a centralized digital platform for patients, doctors, and administrators.
* **Features:**
  + User registration and login with JWT authentication
  + Doctor application and admin approval system
  + Appointment booking and confirmation tracking
  + Role-based dashboards for Users, Doctors, and Admins
  + Real-time status updates
  + Secure authentication and data handling

**3. Architecture**

* **Frontend (React):**  
  Built with React.js, Bootstrap, and Axios. The frontend communicates with the backend through RESTful APIs, handles route-based rendering, and manages state using hooks.
* **Backend (Node.js & Express):**  
  Node.js with Express.js handles API routes, middleware, authentication, and business logic. Protected routes use JWT for security.
* **Database (MongoDB):**  
  MongoDB stores users, doctor applications, and appointment data. Mongoose is used for schema definition and interaction.

**4. Setup Instructions**

* **Prerequisites:**
  + Node.js (v18+)
  + MongoDB (Local or Atlas)
  + VS Code / IDE
  + Git
* **Installation:**

bash

CopyEdit

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

cd docspot

cd server

npm install

cd ../client

npm install

* **Environment Variables:**  
  Create a .env file in the server/ folder with:

ini

CopyEdit

MONGO\_URI=your\_mongodb\_url

JWT\_SECRET=your\_secret\_key

**5. Folder Structure**

* **Client/** (React Frontend)
  + /pages – Dashboard pages
  + /components – Reusable UI components
  + /App.js – Routing and layout
  + /api – Axios config
* **Server/** (Node.js Backend)
  + /models – Mongoose schemas
  + /routes – API endpoints
  + /controllers – Logic handling
  + /middleware – Auth and role protection

**6. Running the Application**

* **Start Frontend:**

bash

CopyEdit

cd client

npm start

* **Start Backend:**

bash

CopyEdit

cd server

npm start

**7. API Documentation**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /api/user/register | POST | Register a new user |
| /api/user/login | POST | Login user and get token |
| /api/doctor/apply | POST | Submit doctor application |
| /api/admin/doctors | GET | Get all doctor applications |
| /api/admin/approve-doctor/:id | PUT | Approve doctor by ID |
| /api/user/appointments | POST | Book an appointment |

*(Add request body & response examples in final Word/PDF)*

**8. Authentication**

* **Method:** JSON Web Tokens (JWT)
* **Password Security:** Bcrypt.js
* **Flow:**
  + On login, backend issues a token
  + Token stored in localStorage
  + All protected routes require a valid token in headers

**9. User Interface**

* Landing Page
* Login/Register
* Apply as Doctor Form
* Admin Approval Dashboard
* Appointment History for Users

**10. Testing**

* **Strategy:** Manual functional testing, API testing
* **Tools:** Thunder Client, Postman, Browser DevTools
* **Scope:** Auth, doctor approval, protected routes, booking, error handling

**11. Screenshots or Demo**

* **GitHub:** <https://github.com/Mitta-Sushma/docspot>

**12. Known Issues**

* No real-time chat or teleconsultation yet
* No email/SMS notifications
* Admin actions are fully manual
* No analytics dashboard implemented

**13. Future Enhancements**

* Payment gateway integration
* Email/SMS notifications
* Real-time chat or video consultations
* Mobile app version
* Doctor ratings & reviews
* Advanced analytics