DocSpot: Seamless Appointment Booking for Health

1. Introduction

Project Title: DocSpot: Seamless Appointment Booking for Health\ **Team Members:**

- Team Leader: Kaitepalli Aditya Krishna
- Team Member: Kakarla Brahmavali
- Team Member: Kandula Sushma

2. Project Overview

Purpose:\ DocSpot is a full-stack MERN application designed to streamline appointment booking in the healthcare sector. It enables users (patients) to easily find and book appointments with doctors based on availability and specialization, while allowing doctors to manage their appointments effectively.

Features:

- User registration and login
- Doctor and patient dashboard
- Booking and managing appointments
- Viewing doctor profiles and availability
- Admin panel for managing users and appointments
- Real-time updates and notifications

3. Architecture

Frontend:

- Built using React.js
- Utilizes functional components and hooks
- Styled with CSS and Bootstrap
- Interacts with backend using Axios for API requests

Backend:

- Built with Node.js and Express.js
- RESTful API endpoints for authentication, appointment handling, and user data
- Middleware for error handling and route protection

Database:

- MongoDB with Mongoose ODM

- Schemas for Users, Doctors, and Appointments
- CRUD operations via Mongoose models

4. Setup Instructions

Prerequisites:

- Node.js (v14 or above)
- MongoDB (local or cloud instance)
- Git

Installation:

1. Clone the repository:

```
git clone https://github.com/KSushma-17/DocSpot.git
```

1. Navigate to the frontend:

```
cd DocSpot/frontend
npm install
```

1. Navigate to the backend:

```
cd ../backend
npm install
```

1. Create a `.env` file in the backend directory with:

```
PORT=5000
MONGO_URI=your_mongodb_connection_string
JWT_SECRET=your_secret_key
```

5. Folder Structure

Client (frontend):

```
frontend/
|- src/
   |- components/
   |- pages/
   |- context/
   |- App.js
   |- index.js
```

Server (backend):

```
backend/
|- controllers/
|- models/
|- routes/
|- middleware/
|- config/
|- server.js
```

6. Running the Application

Frontend:

```
cd frontend
npm start
```

Backend:

```
cd backend
npm start
```

7. API Documentation

Base URL: `/api`

Auth Routes:

- `POST /auth/register`: Register new user
- `POST /auth/login`: Login user

Appointment Routes:

- `GET /appointments`: Get all appointments (admin)
- `POST /appointments`: Book a new appointment
- `DELETE /appointments/:id`: Cancel appointment

Doctor Routes:

- `GET /doctors`: List all doctors
- `GET /doctors/:id`: Get doctor profile

8. Authentication

- JWT-based authentication
- Token stored in HTTP-only cookie
- Protected routes using custom middleware
- Role-based access for Admin, Doctor, and Patient

9. User Interface

The UI features responsive dashboards for patients, doctors, and admins.

- Appointment calendar
- Profile pages
- Forms for booking and managing appointments

Screenshots can be embedded here.

10. Testing

- Manual testing via Postman for all endpoints
- Frontend tested with browser and developer tools
- Future plans for integration tests using Jest and React Testing Library

11. Screenshots or Demo

- Deployed Frontend on Vercel
- Backend API on Render
- Add screenshots of the dashboard and booking flow here

12. Known Issues

- Delay on first API call due to Render free plan cold start
- Limited validation in forms (to be enhanced)
- Mobile responsiveness may need improvements

13. Future Enhancements

- Real-time appointment updates using WebSockets
- Email and SMS notifications
- Payment integration for paid consultations
- Analytics dashboard for admin

- Multi-language support