Project Documentation

1. Introduction

- **Project Title:** DocSpot – Seamless Appointment Booking for Health

2. Project Overview

- **Purpose:**
  The goal of *DocSpot* is to provide a seamless, fast, and user-friendly platform for patients to book appointments with doctors, view schedules, and manage healthcare interactions online.
- **Key Features:**
  - User registration & login (patients and doctors)
  - Browse and book doctor appointments
  - Real-time availability and schedule management
  - Admin panel to manage users and bookings
  - Email notifications for confirmations

3. Architecture

- **Frontend (React.js):**
  - Built with React functional components
  - State management using Context API
  - React Router for navigation
  - Styled with Tailwind CSS or Bootstrap
- **Backend (Node.js + Express.js):**
  - REST API with modular route handlers
  - JWT-based authentication
  - Input validation with express-validator
- **Database (MongoDB):**
  - Collections: `Users`, `Appointments`, `Doctors`, `Schedules`

- Mongoose for schema modeling
- Relationships using ObjectId references

4. Setup Instructions

- **Prerequisites:**
  - Node.js (v18 or above)
  - MongoDB (local or Atlas)
  - Git
- **Installation Steps:**
  - Clone the repo:

```
bash
CopyEdi
t
git clone
https://github.com/yourusername/docspot.git cd
docspot
```

  - Install dependencies:

```
bash
CopyEdi
t
cd client
npm
install
cd ../ser
ver npm
install
```

  - Create `.env` in `/server`:

```
env
CopyEdi
t
MONGO_URI=your_mongo_uri
JWT_SECRET=your_jwt_secret
PORT=5000
```

- Run the app:
  - Client: `cd client && npm start`
  - Server: `cd server && npm start`

5. Folder Structure

- **Client (Frontend):**

```
pgsql
CopyEdi
t
/client
  ├── public/
  ├── src/
  │   ├── components/
  │   ├── pages/
  │   ├── context/
  │   ├── App.js
  │   └── index.js
```

- **Server (Backend):**

```
pgsql
CopyEdi
t
/server
  ├── config/
```

```
├── controllers/
├── middleware/
├── models/
├── routes/
├── utils/
├── server.js
```

6. Running the Application

- **Frontend:**

```
bash
CopyEdi
t


cd
client
npm
start
```

- **Backend:**

```
bash
CopyEdi
t cd
server
npm
start
```

7. API Documentation

Endpoint	Method	Description	Auth Required	Body Params
`/api/auth/register`	POST	Register a new user	No	name, email, password, role
`/api/auth/login`	POST	Login and get token	No	email, password
`/api/doctors/`	GET	Get list of doctors	Yes	—
`/api/appointment/`	POST	Book an appointment	Yes	doctorId, date, time
`/api/admin/users`	GET	Get all users (admin only)	Yes (admin)	—

8. Authentication

- JWT-based authentication
- On successful login, token is stored in localStorage
- Protected routes use `authMiddleware.js` to verify tokens
- Admin routes have additional `roleCheck.js` middleware

9. User Interface

- Home Page with login/register options

- Patient Dashboard – Book appointments, view history
- Doctor Dashboard – Manage availability
- Admin Dashboard – View user stats

Screenshots to be included here or as links to hosted images.

10. Testing

- **Strategy:**
  - Manual UI testing
  - Postman tests for API endpoints
  - Jest for backend logic testing (optional)
- **Tools:**
  - Postman
  - Browser Developer Tools

11. Screenshots or Demo

Link: [Demo Video / Deployed Link]
Screenshots:

- Login Page
- Patient Dashboard
- Appointment Booking Page
- Admin View

12. Known Issues

- Backend server restarts required occasionally on error
- Appointment conflicts not yet automatically resolved
- Notifications not implemented for mobile SMS yet

13. Future Enhancements

- Google/Gmail OAuth login support

- Real-time chat with doctor (Socket.io)
- SMS notifications and WhatsApp integration
- Admin analytics dashboard with charts
- Payment gateway integration (Razorpay / Stripe)