Project Title:

DocSpot: Seamless Appointment Booking for Health

Goal:

To create a full-stack web application where **patients can book appointments**, and **doctors can manage schedules**, using a **secure and responsive** interface.

Features to Implement:

User Roles:

1. **Patient**

   * Sign up / Login
   * Browse doctors by specialization
   * Book appointments
   * View upcoming/past appointments
2. **Doctor**

   * Sign up / Login
   * Manage availability
   * View patient bookings
   * Accept/Reject appointments
3. **Admin (optional)**

   * Manage doctors/patients
   * Approve doctor registrations

Tech Stack (MERN)

Frontend (React.js)

* User Interface (Responsive using Bootstrap/Tailwind)
* React Router for navigation
* Axios for API calls
* Context API or Redux (for auth & global state)

Backend (Node.js + Express.js)

* RESTful API endpoints
* JWT Authentication
* Routes for patients, doctors, appointments

Database (MongoDB)

* Collections:

  * `Users` (with roles: doctor/patient)
  * `Appointments`
  * `DoctorProfiles` (specialty, availability)
  * (Optional) `Admin`

Detailed Steps to Build:

Step 1: Set up the backend

1. Initialize Node.js project:

   ```bash
   npm init -y
   npm install express mongoose dotenv cors bcryptjs jsonwebtoken
   ```

2. Set up Express server:
   `/server/index.js`

3. MongoDB Connection using Mongoose
   `/server/config/db.js`

4. Create models:

   * `User.js`
   * `Appointment.js`
   * `DoctorProfile.js`

5. Create routes:

   * `/api/auth` (register/login for patients and doctors)
   * `/api/doctors` (list doctors, get profile, availability)
   * `/api/appointments` (book, cancel, view)
   * Use **JWT** for authentication middleware.

Step 2: Set up the frontend (React)

1. Initialize React app:

   ```bash
   npx create-react-app docspot-client
   cd docspot-client
   npm install axios react-router-dom
   ```

2. Create pages:

   * `LoginPage`, `RegisterPage`
   * `DoctorDashboard`, `PatientDashboard`
   * `BookAppointmentPage`
   * `DoctorProfilePage`

3. Create reusable components:

   * Navbar
   * AppointmentCard
   * DoctorList

4. Setup Routing:

   * `/login`, `/register`, `/dashboard`, `/book/:doctorId`

5. Auth flow:

   * Store JWT in localStorage
   * Use Axios interceptors for auth headers

Step 3: Appointment Booking Logic

1. Patient selects doctor   views available slots   books appointment
2. Backend checks availability and stores appointment
3. Doctor gets notified (or sees in dashboard)

Step 4: Doctor Availability System

* Each doctor has a profile with:

  * Available days & time slots
  * Specialty
  * Location/contact info

Extra Features (optional but impressive):

* Email confirmations using **Nodemailer**
* Search by location/specialty
* Admin approval for doctors
* Calendar view (React Big Calendar)
* Notifications for appointment status changes

Folder Structure (Backend)

```
server/

    config/
       db.js
    models/
       User.js
       Appointment.js
       DoctorProfile.js
    routes/
       authRoutes.js
       appointmentRoutes.js
       doctorRoutes.js
    middleware/
       authMiddleware.js
    controllers/
       authController.js
       appointmentController.js
       doctorController.js
    index.js
    .env
```

Folder Structure (Frontend)

```
docspot-client/

    src/
       components/
       pages/
       context/
       services/
       App.js
       index.js
```

```

---

##   Final Deployment

* Deploy frontend to **Vercel** or **Netlify**
* Deploy backend to **Render**, **Railway**, or **MongoDB Atlas + Fly.io**
* Set environment variables for production

---

##   Sample `.env` (Backend)

```
MONGO_URI=mongodb+srv://...
JWT_SECRET=yourSecretKey
PORT=5000
```

---

Would you like a **code template**, **ER diagram**, or **API documentation** next?
```