Full Stack Development With MERN:

1.Introduction:

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

Team ID: LTVIP2025TMID46308

Team Size: 4

Team leader: Panthagani Bhuvaneswari

Team member: Rapur Manoj Kumar Reddy

Team member: Chinthapatla Bhargavi

Team member: Shaik Abar Ali

2.Project Overview:

Overview:

Booking a doctor’s appointment is now faster, simpler, and more convenient with our online platform. The system is designed to streamline the process for patients, doctors, and administrators through an intuitive interface and powerful backend.

Purpose:

DocSpot is designed to simplify and modernize the process of booking medical appointments. The goal is to eliminate traditional bottlenecks like long phone waits, manual scheduling, and lack of real-time availability by creating a seamless digital platform for patients, doctors, and administrators.

Features:

🔹 User registration and login with role-based access (Patient, Doctor, Admin)  
🔹 Secure authentication using encrypted passwords  
🔹 Doctor directory with filters for specialty, location, and availability  
🔹 Real-time appointment booking with date and time slot selection  
🔹 Document upload during appointment booking (e.g., reports, insurance)  
🔹 Patients can view, cancel, or reschedule appointments  
🔹 Doctors can confirm, reject, or reschedule appointments  
🔹 Doctor dashboard to manage availability and upcoming appointments

3.Architecture:

**Frontend**

* **Technology Stack**: React.js, HTML, CSS, JavaScript, Bootstrap, Material UI

**Features**:

* + Client-side rendering
  + Responsive and user-friendly interface
  + Axios for RESTful API communication

**Backend**

**Technology Stack**: Node.js, Express.js

**Features**:

* + RESTful APIs for user, doctor, appointment, and admin operations
  + Session and state management
  + JWT-based user authentication
  + Role-based access controls

**Database**

* **Technology**: MongoDB

**Data Storage**:

* + User profiles (Patients, Doctors, Admins)
  + Appointment details (status, date, patient, doctor, documents)
  + Doctor availability and schedules

4.Setup Instructions:

Prerequisites:

* Node.js (v18+ recommended)
* MongoDB (local or cloud via MongoDB Atlas)
* Git

Installation:

Clone the Repository:

git clone [your-github-repo-link]  
 cd project folder

Install Backend Dependencies:

* cd backend  
  npm install

Install Frontend Dependencies:

* cd ../frontend  
  npm install

Create .env File in /backend

* PORT=5000  
  MONGO\_URL=your\_mongodb\_connection\_string  
  JWT\_SECRET=your\_jwt\_secret

5.Folder Structure:

Client(Frontend):

BookADoctor/

├── client/

│ ├── public/

│ └── src/

│ ├── components/

│ │ ├── User/

│ │ ├── Doctor/

│ │ └── Admin/

│ ├── pages/

│ ├── App.js

│ └── index.js

Server(Backend):

├── server/

│ ├── controllers/

│ ├── models/

│ ├── routes/

│ ├── middleware/

│ └── server.js

├── config/

│ └── db.js

├── .env

└── package.json

6.Running the Apllication:

Start the Backend:

cd backend  
npm run dev

Start the Frontend:

cd ../frontend  
npm start

7.API Documentation:

Authentication:

* POST /auth/register — Register user
* POST /auth/login — Login and receive token

Users:

* GET /users/:id — Get user profile
* PATCH /users/:id — Update user info

Doctors:

* GET /doctors — List all doctors
* POST /doctors — Register a doctor (admin access)
* GET /doctors/:id — Get doctor profile

Appointments:

* POST /appointments — Book an appointment
* GET /appointments/:userId — Get patient’s appointments
* PATCH /appointments/:id — Update appointment status

8.Authentication:

 JWT-based authentication

 Role-based access: admin, doctor, user

 Authenticated routes are protected using middleware

 Token sent in headers as Authorization: Bearer <token>

9.User Interface Overview:

Patient Dashboard:

* View appointments
* Book appointment
* Upload medical documents

Doctor Dashboard:

* View upcoming appointments
* Approve or reschedule bookings

Admin Dashboard:

* Approve doctor registrations
* Monitor system usage

Login/Register Pages (with role selection)

10.Testing:

Manual Testing:

* Test all roles (Patient, Doctor, Admin)
* Test appointment flow: create → schedule → cancel
* Test edge cases (invalid input, unavailable slot)

Automated Testing (Optional Setup):

Backend:

* Use Jest or Mocha for API testing
* Example: test POST /appointments route

Frontend:

* Use React Testing Library for component tests
* Cypress for end-to-end testing

11.Demo Drawing & Wireframes:

Home Page  
+---------------------------------------------------+  
| DocSpot Logo | [Login] [Register] |  
|---------------------------------------------------|  
| “Find and book doctors effortlessly.” |  
| [Get Started] |  
+---------------------------------------------------+

Patient Dashboard  
+-------------------------+

| **Sidebar: Home, History** |
| --- |
| Upcoming Appointments |
| [Doctor Name] [Date] |
| [Cancel] [Reschedule] |
| +-------------------------+ |

Doctor Dashboard  
+-------------------------+

| **Sidebar: Schedule, Bio** |
| --- |
| Appointments List |
| [John Doe] [9 AM] [✔️] |
|  |
| +-------------------------+ |

Admin Dashboard  
+------------------------------+  
| New Doctor Registrations |  
| [Dr. Smith] [Approve] [X] |  
| |  
+------------------------------+

12.Known issues:

1.No Email/SMS Notifications:  
  Appointment confirmation and reminders are only shown in-app; no real-time email or SMS is sent.

2. Limited Calendar Integration:  
  Appointment slots do not sync with Google/Outlook calendars.

3.Basic Error Handling:  
  API error responses and form validations are not fully standardized.

4.No Real-time Updates:  
  Appointment status does not update in real-time without refreshing.

5.Limited Role Management:  
  Admin functionalities are minimal and not fully protected via RBAC middleware.

13.Future enhancement:

1.Email & SMS Notification Integration:  
  Use services like SendGrid or Twilio for appointment confirmations and reminders.

2.Calendar Sync:  
  Allow users and doctors to sync appointments with Google or Outlook calendars.

3.Real-Time Notifications:  
  Use WebSockets (Socket.IO) for live updates on appointment status changes.

4.Advanced Search:  
  Implement fuzzy search, auto-suggestions, and location-based filtering.

5.Payment Integration:  
  Enable online payment for bookings using Stripe, Razorpay, or PayPal

14.Additional Information:

1.Tech Stack:  
  Frontend: React.js + Bootstrap/Material UI  
  Backend: Node.js + Express.js  
  Database: MongoDB  
  Auth: JWT-based Role Authentication  
  APIs: RESTful, Axios Integration

2. Development Tools:  
  VS Code, Postman, MongoDB Compass, GitHub

3. Suggested Hosting Platforms:  
  Frontend: Vercel / Netlify  
  Backend: Render / Railway / Heroku  
  Database: MongoDB Atlas

4. Deployment Strategy:  
  Use CI/CD with GitHub Actions or Vercel for auto-deploy

5. Security Notes:  
  Ensure HTTPS, secure cookies, password hashing, and validation before production