**BOOK A DOCTOR USING MERN**

**Team Members:** Aarthy K

Abinesh Bala G

Afrin Fathima P

Ajay P

**OVERVIEW:**

This project is a full-stack web application built using the MERN stack (MongoDB, Express.js, React.js, and Node.js) to provide a user-friendly and efficient system for managing doctor appointments. Patients can search for doctors by specialty, location, or availability, book appointments, view their appointment history, and manage their profile information. Doctors can manage their schedules, view patient information, and update appointment statuses.

**Problem Statement:**

In today’s healthcare system, managing doctor appointments effectively is a critical challenge. Traditional methods of appointment booking, which often involve phone calls or walk-ins, are inefficient, prone to human error, and lead to significant administrative overhead for both doctors and patients. Additionally, patients may struggle to find doctors that match their needs in terms of specialty, location, or availability, leading to frustration and delays in receiving care.

**System Architecture Overview:**

The system architecture follows a **client-server model** with the following main components:

* **Frontend** (Client-side): Built using **React.js**, this handles the user interface (UI), presenting information to the user, and sending requests to the backend.
* **Backend** (Server-side): Built using **Node.js** with **Express.js**, this handles business logic, user authentication, data processing, and database interactions.
* **Database**: **MongoDB** is used to store user data (patients, doctors), appointments, and other system-related information.
* **Authentication & Authorization**: Secure authentication is handled using **JWT (JSON Web Tokens)** to ensure secure access for patients and doctors based on their roles.

**System Workflow:**

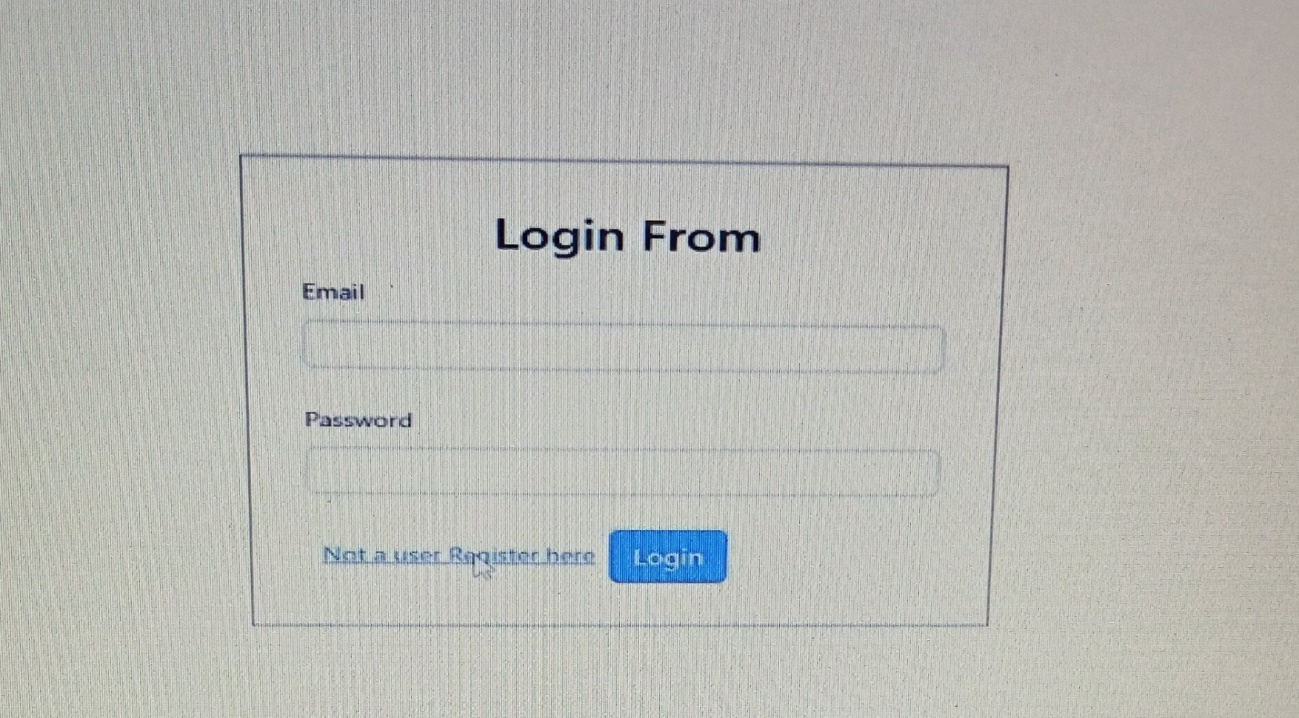
1. **User Registration and Authentication**:
   * Patientsand doctors register by providing essential details, with passwords hashed for security.
   * After registration, users can log in. On successful login, a JWT token is generated and sent to the frontend.
   * The token is stored in localStorage or cookies and used for authenticating future requests.
2. **Appointment Management**:
   * Patients search for doctors by specialty, location, and availability, and bookappointments.
   * Doctors can view patient information, accept/reject appointments, and update their availability.
   * Both parties receive notifications in-app for reminders, updates, or cancellations.
3. **Data Flow**:
   * Frontend (React) sends HTTP requests to the Backend (Node.js/Express) for operations like booking or searching.
   * The Backend interacts with MongoDB to fetch/modify data and responds to the frontend.
   * Frontend dynamically updates the UI by showing confirmation or appointment details.

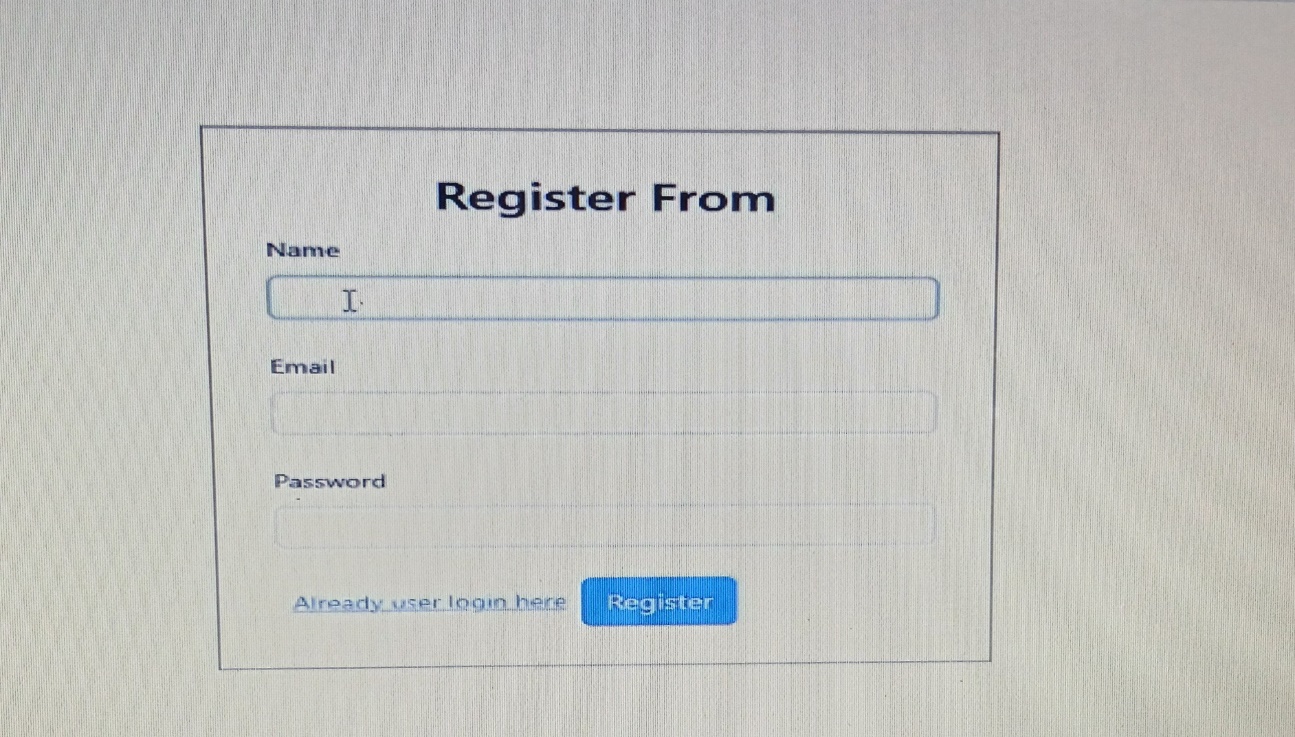
This workflow ensures smooth interaction between patients, doctors, and the backend while maintaining security and efficient data management.

**User Interface Design:**

* **Key Screens**

1. Login/Registration:
   * Simple forms for patient/doctor registration and login.
   * Separate registration for doctors with specialty and availability.





1. Dashboard:
   * Patient: Search doctors, view appointments, and manage profile.
   * Doctor: View schedule, manage appointments, and update availability.
2. AppointmentBooking:
   * Patient: Search for doctors by specialty and location, then book appointments.
   * Doctor: Accept, reject, or reschedule appointments.
3. ProfileManagement:
   * Patient: Update contact details and password.
   * Doctor: Edit specialty, location, and availability.
4. Notifications:
   * Real-time appointment reminders and updates.

* **Responsive Design**

1. Uses Bootstrap grid system for mobile-first design.
2. Collapsiblenavigation for smaller screens.

* **UI Components**

1. Forms for registration, login, and profile editing.
2. Tables and Lists for appointments and doctor information.
3. Buttons and Modals for confirmation actions (Booking/cancelling appointments).

**Testing and Quality Assurance:**

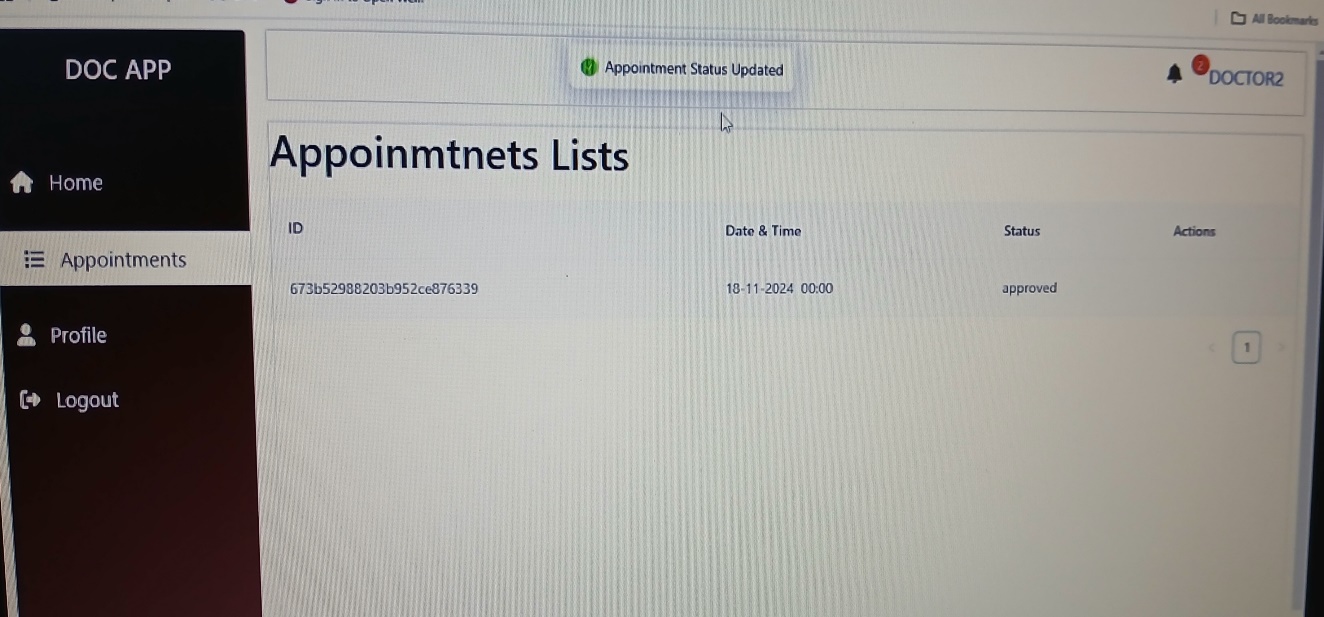
* Types of Testing:
* Unit Testing: Verifies individual components using Jest and React Testing Library.
* Integration Testing: Ensures frontend-backend interaction using Jest and Supertest.
* End-to-End (E2E) Testing: Simulates user actions with Cypress to test entire workflows (e.g., booking an appointment).
* Security Testing: Ensures protection from vulnerabilities using tools like OWASP ZAP and manual penetration testing.
* Performance Testing: Measures system performance under load with Apache JMeter and Lighthouse.
* Manual Testing:
* Verifies core functionality (patient search, booking, doctor profile management).
* Tests edge cases (invalid inputs, unavailable timeslots).
* Bug Tracking and Issue Resolution:
* Bugs are tracked with Jira or Trello, prioritized, and resolved by developers, then retested.
* Test Coverage:
* Jest provides coverage reports to ensure core code is well-tested, aiming for 80-90% coverage.
* CI/CD:
* GitHub Actions or Jenkins automate testing on every pull request.
* Docker ensures consistent testing environments.
* Continuous deployment to Heroku or AWS after passing tests.
* Regression Testing:
* Conducted after updates to ensure new features don’t break existing functionality.
* User Acceptance Testing (UAT):

Real users (patients and doctors) test key features and provide feedback.

* Final Remarks
* Testing ensures the app is secure, functional, and bug-free.
* Ongoing user feedback helps improve future versions.

**Conclusion:**

The Doctor Appointment System is a robust web application that leverages the MERN stack to offer a streamlined and efficient way to manage doctor appointments. With features for both patients and doctors, the system makes it



easier to find, book, and manage medical appointments in a secure and user-friendly manner.

Video Demo link:

https://drive.google.com/file/d/1zOI2DDNVF494Vw5W6YRALq37Cs4CWljV/view?usp=drive\_link