Week 18: DocAppoint - Secure Authentication & Role-Based Access Control

Day 1: Security Strategy and User Authentication

- **The Authentication Engine**: We implemented a stateless authentication system using JSON Web Tokens (JWT) to manage sessions across the frontend and backend.
- **Bcrypt Password Hashing**: To ensure patient data privacy, we integrated the bcrypt library to hash passwords before they are persisted in the MongoDB "Users" collection.
- **Token Issuance**: Upon a successful login attempt, the Express server generates a signed JWT containing the user's unique ID and their specific role (Doctor, Patient, or Admin).

Day 2: Client-Side Security and Session Management

- **Token Storage**: The React frontend captures the JWT and stores it in localStorage to ensure the user remains authenticated across browser refreshes.
- **Protected Routes**: We developed a specialized ProtectedRoute component in React that checks for a valid token and the correct user role before allowing access to sensitive pages.
- **Logout Logic**: Implemented a logout function that clears the local storage and redirects the user to the landing page, effectively ending the session.

Day 3: Role-Based Access Control (RBAC) Middleware

- **Authorization Layers**: Beyond authentication, we created custom Express middleware to verify that a user has the appropriate permissions for specific actions.
- **Doctor-Only Endpoints**: Routes for managing availability and viewing patient lists are protected by an isDoctor middleware that decodes the JWT and validates the role.

- **Admin-Only Endpoints**: Administrative tasks, such as verifying new doctor profiles or managing clinic data, are restricted to users with the "Admin" role.

Day 4: Doctor Portal - Schedule and Profile Management

- **Dashboard Interface**: The Doctor Portal was built as a dedicated dashboard using functional components and the useState hook to manage local UI states.
- **Profile Customization**: Doctors can update their consultation fees, specializations, and bios, which are saved to the "Doctors" collection via PUT requests.
- **Viewing Appointments**: Implemented a real-time list view where doctors can see their upcoming appointments, sorted chronologically using MongoDB query sorting.

Day 5: State Management for Healthcare Portals

- **Session State (Appendix B)**: We utilized global context to track the logged-in user's profile data, making it available to the entire application without redundant API calls.
- **Control State**: Managed the visibility of various UI elements, such as "Success" and "Error" toast notifications, based on the outcome of authentication attempts.
- **Immutability in State**: Ensured all user profile updates followed immutability patterns using the spread operator to prevent side effects in the React lifecycle.