Week 19: DocAppoint - Booking Engine & Time-Slot Logic

Day 1: Theoretical Design of the Booking Engine

- **Real-Time Availability**: The booking engine is designed to handle high-concurrency requests, ensuring two patients cannot book the same time slot.
- **Data Flow**: The process begins with fetching the doctor's available slots from MongoDB and displaying them as interactive buttons on the frontend.

Day 2: Interactive Time-Slot Selection (Frontend)

- **Dynamic Mapping**: We utilized the .map() method to iterate through the availableSlots array and generate clickable slot indicators.
- **Conditional UI Logic**: Slots already occupied are visually disabled and unclickable, utilizing React's conditional rendering based on the slot's "status" property.
- **Selection State**: A local state variable tracks the currently selected slot, enabling the "Book Appointment" button once a valid selection is made.

Day 3: Backend Controller for Appointment Processing

- **Appointment Logic**: The appointmentController was developed to handle the complex logic of verifying slot availability before finalizing the booking.
- **MongoDB Transactions**: To maintain data integrity, the server performs a two-step process: creating the appointment record and updating the doctor's availability status.
- **Response Handling**: The API returns a 201 Created status upon successful booking or a 400 Bad Request if the slot was taken in the interim.

Day 4: Patient Dashboard & Booking History

- **Personalized View**: Patients can access a dashboard showing their "Upcoming Appointments" and "Medical History".
- **Cancelling Appointments**: Implemented a cancellation feature that triggers a DELETE request to the backend and restores the doctor's available slot.
- **Filtering Logic**: Users can filter their history by date range or doctor specialization using frontend filtering and backend query parameters.

Day 5: Communication and Location State (Appendix B)

- **Communication State**: We implemented extensive loading and error states to keep the patient informed during the booking process.
- **Location State Side Effects**: Utilizing the useLocation hook to pass data between the doctor search results and the booking page without re-fetching.
- **Refining the UI**: Added CSS transitions and animations to the booking flow to enhance the user experience of the **DocAppoint** platform.