**Project: Healthcare Management System**

**Description**

Develop a comprehensive healthcare web app for patients, doctors, and administrators. The system should provide features like appointment booking, doctor consultations (video streaming), a medical e-commerce store, patient record management, and a dashboard for admin analytics.

**Features to Include**

**1. Authentication & Authorization**

* **React Topics:** Context API, custom hooks, private routes.
* **Features:**
  + Patient, doctor, and admin roles.
  + JWT-based authentication.
  + Role-based access control (e.g., doctors access only patient records; admins manage users).
* Libraries: react-router, redux-persist, axios.

**2. Appointment Booking System**

* **React Topics:** Forms, state management with Redux, API integration.
* **Features:**
  + Patients can book appointments with doctors based on availability.
  + Appointment calendar view for doctors.
  + Email and SMS notifications (use 3rd-party APIs like Twilio).

**3. Video Consultation (Streaming)**

* **React Topics:** WebRTC or video libraries, handling real-time data.
* **Features:**
  + Live video consultations with doctors.
  + Chat support during the video session.
  + Screen sharing for diagnostics.
* Libraries: peerjs, socket.io-client.

**4. E-commerce for Medicines**

* **React Topics:** Redux (cart management), API integration, forms.
* **Features:**
  + Product catalog with filters (categories, price, availability).
  + Add to cart, remove, and checkout process.
  + Online payment gateway integration (e.g., Stripe, Razorpay).

**5. Patient Records Management**

* **React Topics:** CRUD operations, API calls.
* **Features:**
  + Patients can upload their medical reports.
  + Doctors can view and update patient records.
  + Download reports in PDF format.
* Libraries: react-pdf, axios.

**6. Admin Dashboard**

* **React Topics:** Redux Toolkit (state slice management), charts.
* **Features:**
  + Monitor appointments, sales, and user statistics.
  + Manage users (patients and doctors).
  + Analytics with interactive graphs.
* Libraries: recharts, chart.js.

**7. Responsive Design**

* **React Topics:** Tailwind CSS or Bootstrap.
* **Features:**
  + Mobile-friendly layout.
  + Dashboard, forms, and streaming components optimized for all devices.

**8. Dark Mode & Theming**

* **React Topics:** Context API, custom hooks.
* **Features:**
  + Toggle between light and dark themes.

**9. Global State Management**

* **Redux Topics:** Store, reducers, actions, middlewares.
* **Features:**
  + Manage global states like authentication, e-commerce cart, appointments, etc.

**10. Real-Time Notifications**

* **React Topics:** WebSockets.
* **Features:**
  + Notify patients about their appointment status.
  + Notify doctors of upcoming appointments.
* Libraries: socket.io-client.

**Tech Stack**

1. **Frontend**
   * React.js
   * Redux Toolkit
   * Tailwind CSS / Bootstrap
   * React Router
2. **Backend**
   * Node.js + Express.js
   * MongoDB (or any preferred database)
   * RESTful APIs
3. **APIs**
   * Payment Gateway: Stripe / Razorpay.
   * Video Consultation: WebRTC or Zoom SDK.
4. **Additional Tools**
   * Authentication: JWT.
   * Deployment: Vercel/Netlify for frontend, AWS/Heroku for backend.

**React Topics Covered**

* Functional Components & Hooks (useState, useEffect, useContext, etc.).
* Redux (with Redux Toolkit, middlewares like thunk/saga).
* Routing (protected routes, nested routing).
* API Integration (CRUD operations).
* State Management (local, global, and persisted).
* Styling with CSS frameworks (Tailwind, Bootstrap).
* Form Handling (Formik, Yup).
* Performance Optimization (lazy loading, memoization).
* Testing (Jest, React Testing Library).

**Phases**

1. **Design & Prototyping**:
   * Sketch wireframes for patient, doctor, and admin views.
   * Plan API endpoints.
2. **Setup**:
   * Initialize React, Redux, and backend projects.
   * Configure theming and routing.
3. **Feature Development**:
   * Develop core features iteratively (e.g., authentication, appointment booking).
4. **Testing**:
   * Add unit and integration tests for critical components.
5. **Deployment**:
   * Deploy frontend and backend to live environments.

Part-1

**1. Authentication & Authorization**

* **React Topics:** Context API, custom hooks, private routes.
* **Features:**
  + Patient, doctor, and admin roles.
  + JWT-based authentication.
  + Role-based access control (e.g., doctors access only patient records; admins manage users).
* Libraries: react-router, redux-persist, axios.

>> Patient, doctor, and admin roles.  
  
src/

├── components/ # Reusable components

├── features/ # Redux slices

├── pages/ # Page-level components

├── services/ # API calls

├── store/ # Redux store

├── styles/ # Global styles

└── App.js # Main component