1. React Basics

- **Introduction to React** – What is React? Why use it?
- **JSX (JavaScript XML)** – Syntax extension for JavaScript.
- **Components** – Functional vs. Class Components.
- **Props** – Passing data between components.
- **State** – Managing component data with `useState`.
- **Event Handling** – Handling user interactions.

---

2. React Hooks

- **useState** – Managing component state.
- **useEffect** – Handling side effects.
- **useContext** – Global state management without prop drilling.
- **useReducer** – Alternative to `useState`, useful for complex state logic.
- **useRef** – Accessing DOM elements and persisting values across renders.
- **useMemo & useCallback** – Performance optimizations.

---

3. Component Lifecycle

- **Mounting, Updating, Unmounting** – Understanding how React components update.
- **Class Component Lifecycle Methods** (e.g., `componentDidMount`, `componentDidUpdate`, `componentWillUnmount`).

---

4. React Routing

- **React Router** – Navigation between pages (`react-router-dom`).
- **Dynamic Routing** – Passing URL parameters.
- **Private Routes** – Restricting access to authenticated users.

---

5. State Management

- **Local State** – `useState`, `useReducer`.
- **Context API** – Lightweight state management.
- **Redux** – Centralized state management (`redux-toolkit` for better performance).

- **Recoil, Zustand, Jotai** – Alternative state management libraries.

---

6. Handling Forms

- **Controlled vs. Uncontrolled Components** – Managing form inputs.
- **Form Handling with useState**.
- **Formik & React Hook Form** – Third-party form management libraries.

---

7. API Calls & Data Fetching

- **Fetch API & Axios** – Making HTTP requests.
- **useEffect with Fetch** – Fetching data on component mount.
- **React Query / SWR** – Advanced data fetching with caching and revalidation.

---

8. Performance Optimization

- **React.memo** – Preventing unnecessary re-renders.
- **useMemo & useCallback** – Optimizing expensive calculations and functions.
- **Lazy Loading & Suspense** – Code splitting for performance.
- **Virtualization** – Rendering large lists efficiently with `react-window`.

---

9. UI & Styling in React

- **CSS Modules** – Scoped styles.
- **Styled Components** – CSS-in-JS approach.
- **Tailwind CSS** – Utility-first CSS framework.
- **Material UI, ShadCN, Chakra UI** – Pre-built UI component libraries.

---

10. Error Handling

- **Try-Catch & Error Boundaries** – Handling JavaScript errors in components.
- **Fallback UI** – Displaying custom error messages.
- **React Suspense** – Handling loading states.

---

11. Authentication & Authorization

- **JWT Authentication** – Token-based authentication.
- **OAuth & Social Logins** – Google, Facebook authentication.
- **Protected Routes** – Restricting access based on authentication.

---

12. Testing in React

- **Jest & React Testing Library** – Unit and integration testing.
- **Cypress & Playwright** – End-to-end (E2E) testing.

---

13. Server-Side Rendering (SSR) & Static Site Generation (SSG)

- **Next.js** – Hybrid SSR & SSG framework.
- **Gatsby** – Static site generator using React.

---

14. React with Backend Technologies

- **Node.js & Express.js** – Building a backend API for a React frontend.
- **GraphQL with React (Apollo Client)** – Alternative to REST APIs.
- **Firebase & Supabase** – Backend-as-a-service solutions.

---

15. Advanced React Concepts

- **Custom Hooks** – Reusable logic.
- **Portals** – Rendering outside the main DOM tree.
- **Higher-Order Components (HOCs)** – Reusable component logic.
- **Render Props** – Alternative component composition pattern.
