I. React Fundamentals & Core Concepts

• What is React and why is it used? (Focus on its purpose as a UI library, component-based architecture, and declarative nature.)

• What are the main features of React? (e.g., JSX, Virtual DOM, Components, Unidirectional Data Flow, Hooks.)

• Explain the concept of Components in React. What are their characteristics?

• What is JSX? Why do we use it, and how does it relate to regular JavaScript?

• Explain the Virtual DOM. How does React use it to improve performance during updates?

• What are Props in React? Are they mutable or immutable, and why?

• What is State in React? How is it different from Props?

• What is the primary difference between props and state?

• What are React Fragments? When would you use them?

• What is ReactDOM used for?

• Explain the significance of keys in lists. What problems do they solve, and what happens if you don't use them or use an index?

• What is conditional rendering in React? Describe various ways to implement it.

• What are synthetic events in React? How do they differ from native browser events?

II. Hooks & Component Lifecycle

• What are React Hooks? Why were they introduced, and what problems do they solve?

• Explain the purpose and common use cases of useState. How do functional updates (setState(prev => ...)) work?

• Explain the purpose and common use cases of useEffect. How do you replicate componentDidMount, componentDidUpdate, and componentWillUnmount using it? What is the role of the dependency array?

• When would you use useReducer instead of useState?

• What is useRef used for? How is it different from useState? Provide examples.

• What is useContext and why do we use it? When is it a good alternative to prop drilling, and what are its limitations?

• Explain useMemo and useCallback. What problems do they solve, and when should they be used for performance optimization?

• What is the difference between useEffect and useLayoutEffect? When would you choose one over the other?

• What are Custom Hooks? How do you create one, and what are their benefits?

• What are the rules of Hooks? Why are they important?

III. Advanced React Concepts

• Explain the Reconciliation process in React. How does React determine what parts of the DOM to update?

• What is React Fiber? How did it improve React's architecture and rendering process?

• How does React handle reconciliation when keys are involved in lists?

• What are Controlled and Uncontrolled Components? When would you use each for form inputs?

• Explain 'lifting state up' in React. When is it a good practice?

• What is a Higher-Order Component (HOC)? How does it work, and what are its pros and cons?

• What is the Render Props pattern? How does it compare to HOCs?

• What is React.lazy() and Suspense used for? How do they facilitate code splitting?

• What are Error Boundaries in React? How do you implement them, and what kind of errors do they catch/not catch?

• What is a React Portal? When would you use it (provide specific examples)?

• How does Server-Side Rendering (SSR) work with React? Explain its benefits (SEO, performance) compared to Client-Side Rendering (CSR).

• What is Hydration in React? How does it relate to SSR?

• Briefly explain Concurrent Mode (or Concurrent Features in React 18). What problems does it aim to solve?

• What are useTransition and useDeferredValue? (Relevant for React 18+)

IV. Performance & Optimization

• How can you optimize the performance of a React application? (Comprehensive answer covering bundle size, rendering, network, etc.)

• What is React.memo()? When should you use it, and what are its limitations?

• What is Code Splitting in React? How does it improve load time?

• How do you identify performance bottlenecks in a React application? (Tools like React DevTools Profiler, Lighthouse).

• What are the Core Web Vitals (LCP, FID, CLS) and how do they relate to React application performance?

V. Ecosystem & Best Practices

• Explain Prop Drilling and how it can be avoided. (Discuss Context API, state management libraries).

• What techniques do you use for state management in large React applications? (Discuss Context, Redux/RTK, Zustand, React Query, etc., and when to choose which).

• What is Redux and why is it used? What are its core principles (Store, Actions, Reducers, Dispatchers)?

• What is Redux Thunk (or Redux Saga) and why is it used?

• How is React integrated with TypeScript? What are the benefits of using TypeScript in a large React project?

• How do you handle routing in React applications? (Focus on React Router). How can you navigate programmatically?

• How can you test React components? What are the common testing libraries/approaches (e.g., Jest, React Testing Library, Enzyme)?

• What are some best practices for building scalable and maintainable React applications? (Component structure, folder organization, clean code, accessibility, error handling).