React and Redux Topics and Questions

React Core Concepts

Theory

- What is React.js?
- What are the advantages and disadvantages of React.js?
- What is the difference between a library and a framework?
- What is a single-page application (SPA)?
- What is the difference between single-page and multi-page applications?
- Why is React called a single-page application?
- What is the purpose of `index.html` in a React SPA?
- What is the Virtual DOM?
- What is the difference between the Real DOM and Virtual DOM?
- What is the Shadow DOM?
- What is the difference between Shadow DOM and Virtual DOM?
- What is reconciliation?
- What is the diffing algorithm?
- What is React Fiber?
- What is the purpose of the root DOM element in React?
- What is JSX?
- Can we write JavaScript code inside JSX?
- Is JSX mandatory to create React.js apps?
- What is the difference between JSX and HTML?
- What are the rules of JSX?
- What is the purpose of `htmlFor` attribute in JSX?
- What is `ReactDOM.createRoot`?
- What is `React.createElement`?
- What is `React.cloneElement`?
- What is a component in React?
- What is the difference between class components and functional components?
- What are stateless vs stateful components?
- What are pure components?
- What is `React.memo`?
- What is the difference between `React.memo` and pure components?
- What is `React.StrictMode`?

- How does enabling/disabling `StrictMode` affect React?
- What are built-in components in React (e.g., Fragment, Suspense, StrictMode)?
- What are React Fragments?
- What are the advantages of React Fragments over `<div>`?
- What are the drawbacks of using `<div>` instead of React Fragments?
- What is one-way data binding?
- What are synthetic events in React?
- What are the advantages of using synthetic events?
- What is event pooling in React?
- What are examples of synthetic events?
- What is `e.preventDefault` and its purpose?
- What is a transpiler?
- What is Babel?
- What is Webpack?
- What is the difference between `package.json` and `package-lock.json`?
- What is Vite?
- What are the advantages of Vite over Create React App (CRA)?
- What is `vite.config.js`?
- What is hot module replacement?
- What are ESModules?
- What are polyfills?
- What is hydration?
- Why is React not SEO-friendly?
- What is the difference between client-side rendering (CSR) and server-side rendering (SSR)?
- Why is SSR faster than CSR for initial loading?
- How does initial load time differ between CSR and SSR?
- What are the drawbacks of CSR?
- What are the benefits of CSR?
- What is code splitting in React?
- What is lazy loading in React?
- What is `React.lazy` and how is it used for code splitting?
- What is Suspense and fallback in React?
- What is the disadvantage of lazy loading?
- What is tree shaking in React?
- What is HTML sanitization?
- What is `dangerouslySetInnerHTML`?
- What is the `span` tag?
- What is `useId` in React?
- What are React portals?
- What is incremental rendering?
- What is React concurrent mode?
- What is automatic batching in React?
- What are the limitations of functional components?

- What are React Mixins?
- What is the difference between `png` and `jpg`?
- What are vector images?
- What is the difference between decryption and decoding?
- What is base64 encoding?
- What is the purpose of keys in React?
- Why shouldn't we use the index of an array as a key prop?
- Can we use the index of an array as a key prop?
- What is component composition?
- What is `props.children`?
- What is props immutability?
- What is prop drilling?
- What is the disadvantage of prop drilling?
- How to prevent prop drilling?
- What is lifting state up in React?
- What is state in React?
- What is the difference between state and props?
- What is the difference between state and local variables?
- What is the difference between state variables and normal variables (e.g., `let`, `const`)?
- Why are arrays and objects copied when updating state?
- What is the purpose of using the spread operator when updating state?
- What is the immutability of state and why is it important?
- What is referential equality?
- What is the purpose of using curly braces around named imports?
- What is the `as` keyword in imports?
- What is destructuring in React?
- Why does a console message in `App.jsx` appear twice in the console?
- What are error boundaries?
- How do error boundaries work in React?
- What is `componentDidCatch`?
- What is method binding in React?
- What is `super()` and `super(props)` in class components?
- What is `forceUpdate()`?
- What is `replaceState()`?
- What are lifecycle methods in React?
- What are lifecycle methods in class components?
- What are lifecycle methods in functional components?
- How to handle `componentDidMount` in functional components?
- How to handle `componentDidUpdate` in functional components?
- How to handle `componentWillUnmount` in functional components using `useEffect`?
- How do lifecycle events work with `useEffect`?
- What are different ways to re-render a React component?
- What is dynamic rendering?

- How does conditional rendering work in React?
- What are the ways to perform conditional rendering (e.g., using `&&`, ternary operators)?
- What is the flow of React?
- How does React work?
- How does React update the actual DOM?
- Why are two Virtual DOMs needed in React?
- What is DOM direct manipulation?
- How does SPA load dynamic data?
- How to integrate React into an existing application?
- What are web workers?
- What is the purpose of the `render()` method in class components?
- What is React profiler?
- What is profiling in React?
- How to debug a React application?
- How to improve the performance of a React application?
- What are optimization techniques in React?
- How to load millions of data smoothly from server-side to client-side?
- What are the latest features in React?
- What is React 19?
- What are directives (`use client`, `use server`) in React?
- What is the difference between `map`, `filter`, and `reduce` methods?

Practical

- Create a component to add two numbers by clicking a button.
- Create a timer using `useEffect`.
- Implement a counter using `useState` or `useReducer`.
- Check whether text entered in two input fields matches.
- Create an input field and display entered text in an `<h1>` tag.
- Change the background color using `useRef`.
- Change the color of a `<div>` when clicking a button.
- Change the text color dynamically using `useRef`.
- Set a timer with start and stop functionality using `useRef`.
- Re-render a component when the page is resized.
- Show fields in an array of objects using `map`.
- Add a style to each title field in a list.
- Create a toggle button (on/off).
- Implement conditional rendering using `&&` syntax.
- Create a component to find the square of a number using `useMemo` (avoid recalculation for the same value).
- Display the current count of todos in a todo list.
- Display the count of completed todo items in a todo list.
- Add an edit feature to a todo app.
- Add a completed button to mark a task as completed in a todo app.

- Prevent adding duplicate tasks in a todo list.
- Add validations for a todo app (e.g., empty, spaces, capital, duplicate).
- Save todos in local storage.
- Re-order todo items and display when each todo was added (e.g., "1m ago", "1h ago").
- Implement a todo list using `useReducer`.
- Create a calculator using `useContext`.
- Implement an increment and decrement counter using `useContext`.
- Create a component to add an item to a list by clicking a button using Redux.
- Implement a counter using Redux.
- Fetch data from an API using `useEffect`.
- Avoid using static values in code.

Hooks

Theory

- What are hooks in React?
- What are the rules of hooks?
- Why can't hooks be used inside conditionals?
- What is `useState`?
- What is the return data type of `useState`?
- How to update state that's an array?
- What is `useEffect`?
- What is the purpose of `useEffect`?
- What happens if no dependency array is passed in `useEffect`?
- What is the behavior of `useEffect` when the dependency array is empty vs. not provided?
- What is the cleanup function in `useEffect`?
- Why can't we use `async` directly in `useEffect`?
- How to handle mounting, updating, and unmounting using `useEffect`?
- What is `useLayoutEffect`?
- What is the difference between `useEffect` and `useLayoutEffect`?
- What is `useRef`?
- What are the advantages of `useRef`?
- Can we store data inside `useRef`?
- How to update `useRef` (e.g., `ref.current.value`)?
- What is the difference between `useRef` and a regular variable?
- What is `useCallback`?
- When should you use `useCallback`?
- What is `useMemo`?
- What is the difference between `useMemo` and `useCallback`?
- What is the difference between `useMemo` and `React.memo`?
- Does `useMemo` prevent re-renders?

- What happens if the dependency array is wrong in `useMemo` or `useCallback`?
- What happens if an empty dependency array is passed to `useMemo` or `useCallback`?
- What is `useContext`?
- What are the use cases of `useContext`?
- What are the limitations of `useContext`?
- Can a component have multiple `useContext` calls?
- What is `useReducer`?
- What is the difference between `useState` and `useReducer`?
- What is the return value of a reducer function?
- What is the third argument in `useReducer`?
- What is `useNavigate`?
- What is the difference between `useHistory` and `useNavigate`?
- What is `useLocation`?
- What is `useImperativeHandle`?
- What is `useTransition`?
- What is `useFormState`?
- What is a custom hook?
- What are the rules of custom hooks?
- What is the difference between a hook and a function?
- What is the purpose of the callback function in `setState`?
- What is stale closure?

Practical

- Create a custom hook.
- Create a custom hook for an increment/decrement counter.
- Use `useEffect` to handle lifecycle events.
- Use `setInterval` and `clearInterval` in a `useEffect`.
- Use `useCallback` in a component.
- Use `useMemo` in different scenarios.
- Implement a counter using `useContext`.
- Implement a counter using `useReducer`.
- Use `useRef` to communicate between components.
- Send data from child to parent using `useRef`.
- Create a timer using `useEffect`.
- Fetch data from an API using `useEffect`.
- Use `useContext` to save and render user data in components.
- Replace `useState` with `useReducer` in a component.

Component Communication

Theory

- What is parent-to-child communication in React?
- What is child-to-parent communication in React?
- How to pass data from parent to child component?
- How to pass data from child to parent component?
- How to pass state from parent to child and vice versa?
- How to pass props from child to parent using callback functions?
- How to update a child component's state from a parent component?
- What is sibling communication in React?
- How to send data from child to parent using `useRef`?
- How to send data from child to parent using the Context API?

Practical

- Pass data from parent to child component.
- Pass data from child to parent component using props.
- Pass data from child to parent using callback functions.
- Pass data from child to parent using `useRef`.
- Pass data from child to parent using the Context API.
- Implement sibling communication using middleware.

State Management

Theory

- What is state management in React?
- What is the Context API?
- What is the difference between Context API and Redux?
- What is `React.createContext`?
- What are Provider and Consumer in the Context API?
- How to update data in the Context API?
- How to modify state passed down through context?
- How to implement context for parsing the user logged-in state?
- What is the difference between local storage and Context API?
- What is the difference between local storage and session storage?
- What are the use cases of browser storage (e.g., localStorage, sessionStorage)?
- What is `localStorage` vs Redux?

Practical

- Save user data in the Context API and use it in components.
- Update data in the Context API.

- Implement a counter using the Context API.
- Store items in local storage.
- Implement context for parsing the user logged-in state.

Routing

Theory

- What is React Router?
- What are the types of routers in `react-router-dom` (e.g., BrowserRouter, HashRouter, MemoryRouter)?
- What is the difference between BrowserRouter and MemoryRouter?
- What is the difference between `<Link>` and `<a>`?
- What is the difference between `Link` and `NavLink`?
- What is an Outlet in React Router?
- What is `useNavigate`?
- What is the difference between `useHistory` and `useNavigate`?
- What is `useLocation`?
- What are the parts of a URL?
- How to read the current URL?
- How to read query parameters and path parameters?
- What is dynamic routing?
- What is nested routing?
- How to redirect a user in React Router?
- How to protect routes in React?
- How to restrict auth pages once the user is authenticated?

Practical

- Read query parameters and path parameters.
- Implement dynamic routing.
- Implement nested routing.
- Protect routes using authentication.
- Redirect to the login page if the user is logged in.
- Use environment variables to store secrets.

Higher-Order Components (HOCs)

Theory

- What are higher-order components (HOCs) in React?
- How do HOCs work?
- What are the advantages of using HOCs?

- What are use cases of HOCs?
- What is a props proxy for HOCs?

Practical

- Create a higher-order component.
- Create a props proxy for an HOC.
- Implement an example for a higher-order component.

Error Handling

Theory

- What is error handling in React?
- What are error boundaries?
- How to implement error boundaries?

Practical

- Implement an error boundary in a React application.
- Show proper error messages (e.g., for form validation, unique email).

Controlled and Uncontrolled Components

Theory

- What are controlled components?
- What are uncontrolled components?
- What is the difference between controlled and uncontrolled components?
- What are the benefits of uncontrolled components?
- How to pull data from an uncontrolled component?
- How to access data from an uncontrolled component?
- How to manipulate uncontrolled components?

Practical

- Create a controlled component.
- Create an uncontrolled component.
- Pull data from an uncontrolled component.

Refs

Theory

- What is `useRef`?
- What are the advantages of `useRef`?
- What is the difference between `useRef` and `createRef`?
- What is `forwardRef`?
- What is the difference between `useRef` and `forwardRef`?
- What is `useImperativeHandle`?
- What is the difference between `useRef` and state?

Practical

- Use `useRef` to change the text color.
- Use `useRef` to set a timer with start and stop functionality.
- Use `forwardRef` in a component.
- Find the average of two numbers using `useRef`.
- Communicate between components using `useRef`.

Event Handling

Theory

- What are event handlers in React (e.g., `handleClick`, `handleChange`)?
- What is event handling in React?
- What is a synthetic event?
- What is event pooling in React?
- What are examples of synthetic events?
- What is `e.preventDefault`?

Practical

- Implement event handlers (e.g., `onClick`, `onChange`).
- Implement a toggle button.
- Implement a list using `<ul>` and `<li>`.

Performance Optimization

Theory

- How to improve the performance of a React application?
- What are memoization techniques in React?
- What is the purpose of `useMemo`, `useCallback`, and `React.memo`?

- What are the limitations of `useMemo` and `useCallback`?
- What is code splitting?
- What is lazy loading?
- What is `React.lazy`?
- What is Suspense and fallback?
- What is tree shaking?
- What is debouncing?
- What is throttling?
- What is the disadvantage of memoization?
- What is starvation in React?
- What are optimization techniques for a React app?
- How to handle memory leaks in React?

Practical

- Implement lazy loading (e.g., load a component after 5 seconds).
- Use `React.lazy` for dynamic component loading.
- Implement debouncing.
- Implement throttling.
- Create a component to find the square of a number using `useMemo`.

Redux

Theory

- What is Redux?
- What are the core principles of Redux?
- What is the Flux architecture?
- What is the difference between Redux and Flux?
- What is the difference between Context API and Redux?
- Why is Redux unidirectional data flow?
- What is a Redux store?
- What is the role of a store in Redux?
- What are Redux store methods (e.g., `getState`, `dispatch`, `subscribe`)?
- What are actions in Redux?
- What are action creators?
- What is a reducer in Redux?
- Why is a reducer a pure function?
- What is `combineReducers`?
- What is the purpose of the Provider component in React-Redux?
- What is `mapStateToProps`?
- What is `mapDispatchToProps`?
- What is the difference between `mapStateToProps` and `useSelector`?

- What is the `connect` function in Redux?
- What are selectors in Redux?
- What is the purpose of selectors in Redux?
- What are Redux middlewares?
- What is the purpose of middleware in Redux?
- What is Redux Thunk?
- What is Redux Saga?
- What is the difference between Redux Thunk and Redux Saga?
- What is `createAsyncThunk`?
- What is `applyMiddleware`?
- What is `call()` and `put()` in Redux Saga?
- What is Redux Toolkit?
- What is `configureStore` in Redux Toolkit?
- What is a Redux slice?
- What are the benefits of using slices in Redux?
- What are constants in Redux?
- What is `extraReducers` in Redux Toolkit?
- What is Redux Persist?
- How does Redux Persist work?
- Why is immutability important in Redux, and how is it achieved?
- What is Immer, and how is it used in Redux?
- How to handle asynchronous actions in Redux?
- How to handle impure actions in Redux?
- Where should API calls be made in Redux?
- What are the disadvantages of Redux?
- What are the pros and cons of Redux?
- What is the difference between `useReducer` and Redux?
- Why can't we use `localStorage` instead of Redux?
- What are the limitations of Redux?
- How does Redux ensure components are updated correctly?
- How to structure a Redux application for large-scale projects?
- How to test Redux reducers?
- What is Redux DevTools?
- How to debug a Redux application?
- What are common performance optimizations for Redux applications?
- What is the Redux folder structure for large-scale applications?

Practical

- Create a todo app using Redux Toolkit.
- Implement a counter using Redux.
- Implement a reducer to handle multiple action types.
- Create an action in Redux.
- Dispatch an action with a payload.

- Use `useSelector` and `useDispatch` hooks.
- Implement `mapStateToProps` and `mapDispatchToProps`.
- Save user email in the Redux store and render it in a component.
- Update data in the Redux store by dispatching an action.
- Implement `combineReducers`.
- Use Redux DevTools.
- Implement `createAsyncThunk`.
- Add middleware in Redux.
- Write a middleware to log a JWT token.
- Implement pagination in a user list.
- Move user list search to a backend API.
- Clear Redux state.
- Implement Redux Thunk.
- Implement Redux Saga.

JWT and Authentication

Theory

- What is a JWT?
- What is the structure of a JWT?
- What are JWT claims?
- What is the JWT signature?
- What is the use of a secret key in JWT?
- What is the use of the signature in JWT?
- What is `jwt.verify` vs `jwt.decode`?
- What are the arguments for `jwt.verify`?
- How does the server validate a JWT?
- What is the security offered by JWT?
- What is the purpose of access tokens and refresh tokens?
- What determines the lifetime of an access token?
- What is the difference between access tokens and refresh tokens?
- Where should access and refresh tokens be stored on the client side?
- What is the ideal place to store a JWT token?
- How to detect token expiry in a frontend application?
- How to update an expired JWT token?
- What is the need for specifying "Bearer" in a token?
- What are sessions vs tokens?
- What are HTTP-only cookies?
- Why doesn’t logout remove cookies?

Practical

- Protect admin URLs using JWT.
- Handle token expiry in a frontend application.
- Implement JWT generation and verification code.
- Write a middleware to console a JWT token.

Routing and Navigation

Practical

- Read path parameters.
- Read query parameters.
- Implement dynamic imports.
- Use `useLocation` in a component.
- Restrict auth pages once the user is authenticated.
- Understand changes when a user logs out.

Form Handling

Practical

- Implement signup and login forms in an OLX clone.
- Add proper form validations (e.g., email uniqueness, file type, file size).
- Show proper validation messages.
- Ensure all input fields are validated.
- Implement item input validation for an OLX app.
- Prevent trimming passwords.

Miscellaneous

Theory

- What is the difference between `fetch` and `Axios`?
- What are Axios interceptors?
- What is an Axios cancel token?
- What is the difference between a transpiler and a compiler?
- What is PropTypes in React?
- How to validate props using PropTypes?
- What is `Props.children`?
- What is componentDidCatch?
- What is the viewport in React?
- What is IndexedDB?

- What are dev dependencies in `package.json`?
- Why do we use PascalCase for component names in React?
- What is JIT (Just-In-Time) compilation?
- What is dead code elimination?

Practical

- Explore Axios instances and interceptors.
- Use `.env` files to store credentials or tokens.
- Avoid loose API calls in components by creating an Axios instance.
- Implement a search feature.
- Add a product and redirect to the product details page.
- Ask for confirmation before deleting an item.
- Suppress unwanted error messages.
- Add logging using logging libraries.
- Create a don’t-have-permission page.
- Manage sessions and cookies.
- Implement optimistic updates.
- Ensure the app is mobile-optimized.
- Clone 4-5 pages for an OLX project.
- Add edit/delete user functionality on the admin side.
- Ensure the user cannot access the dashboard after account deletion by admin.
- Display when a todo item was added (e.g., "1m ago").
- Avoid browser alerts in the app.