Scenario base Practice Question

Created	@October 16, 2025 1:42 PM
Tags	

Component Design & State Management

Scenario: You have a product listing page where filters (price, category, rating) are applied. When the user goes back from product detail to listing, the filters reset.

How would you persist the filter state so that users don’t lose their selection?

Scenario: You need to build a form with 20+ fields, nested objects, and dynamic validations.

Would you choose controlled or uncontrolled components? Why? How would you structure validation to avoid performance bottlenecks?

Scenario: You have a parent component with heavy child components. Even if only one prop changes, all children re-render.

How do you prevent unnecessary re-renders? Which approach would you prefer: React.memo, useMemo, or useCallback?

Performance & Optimization

Scenario: In a dashboard, you’re fetching large data (10k+ records). The UI is laggy when rendering the table.

How would you optimize rendering? (e.g., virtualization, pagination, lazy loading, etc.)

Scenario: A page shows live stock price updates every second using WebSocket. After some time, performance drops.

What would you check first? How do you optimize re-renders for such real-time data?

API & Data Handling

Scenario: You are calling multiple APIs on page load. Some APIs are independent, while others depend on previous API results.

How would you structure the calls? Would you use Promise.all, async/await, or state management tools?

Scenario: Suppose a POST API request fails due to network issues. The user expects offline retry once the network is back.

How would you implement this retry mechanism?

Routing & Navigation

Scenario: A user is filling a multi-step form. If they refresh the page in the middle, they lose progress.

How would you persist the progress? Would you use localStorage, Redux-persist, or something else?

Scenario: You want to prevent users from leaving a page with unsaved form data.

How would you handle this in React Router?

Real-world Challenges

Scenario: You integrate a third-party component (say a chart library) that directly manipulates the DOM. It breaks when React re-renders.

How would you handle this integration safely?

Scenario: Your app has both **desktop and mobile UIs**, and you need to optimize bundle size for performance.

How would you handle code splitting and conditional imports?

Scenario: After deploying your React app, users report that the first page load is too slow.

What steps would you take to analyze and improve performance? (bundle analyzer, lazy loading, caching, etc.)