National University of Computer and Emerging Sciences

Laboratory Quiz 3

for

Web Engineering (SL3003)

Lab Instructors	Sana Ejaz, Hassan Raza
Section	6C 1,2
Semester	Spring 2025

Department of Software Engineering

FAST-NU, Lahore, Pakistan

React Concepts Quiz (90 Minutes)

Instructions:

- Read each part carefully before coding.
- Implement the required functionality step-by-step.
- Focus on correctness and understanding of the React concepts involved.
- You may use JSONPlaceholder's /users endpoint for data: `https://jsonplaceholder.typicode.com/users`

Scenario:

You will build a small "User Directory" application. It will fetch a list of users, display them in cards, allow searching/filtering by name, and include a small interaction on each user card. Performance optimization is required.

Part 1: Fetching and Displaying Users (useEffect, useState, Axios, Components)

1. **Main Component:** Create a functional component named UserDirectory.
2. **State:** Set up state within UserDirectory to manage:
   - The list of users fetched from the API.
   - A loading status indicator.
   - A potential error message.
3. **Data Fetching:** Use the useEffect hook to fetch data from the /users API endpoint *only once* when the UserDirectory component mounts. Use the Axios library for the request.
   - Update the state accordingly based on the fetch outcome (success or failure).
   - **Hint:** Remember how to make useEffect run only on mount.
4. **Basic Display:**
   - While loading, display a simple "Loading users..." message.
   - If an error occurs, display the error message.
   - Once data is loaded successfully, display the name of each user in a simple list (ul/li).

Part 2: User Card Component (Components, Props)

1. **Child Component:** Create a new functional component named UserCard.

2. **Props:** Modify UserCard to accept a user object as a prop. Inside UserCard, display the user's name and email. You can structure this simply (e.g., using divs or p tags).
3. **Refactor Display:** Modify UserDirectory from Part 1. Instead of rendering simple list items with names, map over the fetched users array and render a UserCard component for each user, passing the entire user object as a prop to UserCard.

Part 3: Controlled Search Input (useState, Controlled Components)

1. **Search State:** Add a new piece of state to UserDirectory to store the current search term entered by the user.
2. **Input Element:** Add a text input field within the UserDirectory component (e.g., above the list of user cards).
3. **Controlled Component:** Make this input a controlled component:
   - Its value should be bound to the search term state variable.
   - Its onChange event handler should update the search term state variable whenever the user types.
4. **(No filtering logic needed *yet* - just the controlled input)**

Attempt either Part 4 or 5. You are NOT required to attempt both of the following parts.

Part 4: Filtering Logic & Optimization (useMemo)

1. **Filtering:** Implement the logic to filter the displayed users. Based on the searchTerm state from Part 3, create a *new* list containing only users whose name (case-insensitive) includes the searchTerm.
2. **Optimization:** Filtering can be computationally intensive. Use the useMemo hook to ensure this filtering calculation *only* re-runs when the original list of users changes *or* when the searchTerm changes.
3. **Render Filtered List:** Update UserDirectory to map over and render UserCard components based on this *memoized, filtered list* instead of the original full list of users.

Part 5: Interaction & Callback Optimization (useCallback)

1. **Interaction Button:** Add a simple button inside the UserCard component, labelled "Log User ID".
2. **Handler Function:** Define a function inside UserDirectory called handleLogUserId which accepts a userId as an argument and logs a message like "User ID: [userId]" to the console.
3. **Pass Handler as Prop:** Pass the handleLogUserId function down as a prop (e.g., named onLogId) from UserDirectory to each UserCard.
4. **Connect Button:** In UserCard, make the "Log User ID" button call the received onLogId prop function, passing the specific user's id when clicked.
5. **Callback Optimization:** Wrap the definition of handleLogUserId inside UserDirectory with the useCallback hook. Ensure its reference stability by providing the correct dependencies (if any). Think about why this is important when passing functions to multiple child components.