SWT 22022: Practical for Internet Application Development
Department of Information & Communication Technology
Faculty of Technology
South Eastern University of Sri Lanka

Lab Sheet No: 14

Date: - 2026.01.08

Title: React Routes and API

Aims:

- Creating a dynamic list
- Using context to share data between components
- Building a simple routing system
- API and Axios

Tasks:

- Use **useState** to manage list
- Use **Context API** to pass data between components without props drilling.
- Use **React Router** for client-side routing
- Use **Axios** to fetch data from an API

Activities

1. Managing a Dynamic List using useState.
    - Create a **React functional component** named **DynamicList** that uses the **useState** hook to manage a list of items.
    - Initialize a state variable called **items** as an **empty array** using the **useState** hook.
    - Create a function named **addItem** that:
        - Adds a new item to the list when called
        - The item should be named in the format: **Item 1, Item 2, Item 3, ...**
        - Use the **spread operator** to update the state.
    - Add a **button** to the component that:
        - Displays the text "**Add Item**"

    - Calls the **addItem** function when clicked.
- Display the list of items using:
    - An unordered list (**<ul>**)
    - The **map()** function to render each item inside a **<li>** element
    - Use **index** as the **key** for each list item.
- Export the **DynamicList** component and import it into **App.jsx**.
- Render the **DynamicList** component inside the **App** component so that clicking the button dynamically adds items to the list.

2. Using Context API to Share Data Without Props Drilling.
    - Create a Context named **MyContext** using the **createContext()** function from React.
    - Create a Context Provider component named **ContextProvider** that:
        - Accepts **children** as props
        - Uses the **useState** hook to store shared data with an initial value of "**Initial Data**"
    - Provide the state variable **sharedData** and the updater function **setSharedData** to all child components using the **MyContext.Provider**.
    - Wrap the **children** components inside the **MyContext.Provider** so that all nested components can access the shared data.
    - Create a component named **ComponentNew** that:
        - Uses the **useContext** hook
        - Consumes **MyContext** to access **sharedData** and **setSharedData**
    - Display the value of **sharedData** inside a paragraph (**<p>**) element in **ComponentNew**.
    - Add a button in **ComponentNew** that:
        - Displays the text "**Update**"
        - Updates the shared data to "**Updated Data**" when clicked
    - Export the **ContextProvider** and **ComponentNew** components so they can be used in other parts of the application.
    - Update **App.jsx** to wrap **ComponentNew** inside the **ContextProvider** so that the shared context data is available and the output is displayed correctly in the browser.

3. Using React Router for client-side routing

- Install **React Router DOM** in a React project using the appropriate **npm command**.
- Create a functional component named **Home** that displays the heading "**Home**".
- Create a functional component named **About** that displays the heading "**About**".
- Create a routing component named **AppRoutes** that:
  - Imports **Routes** and **Route** from **react-router-dom.**
  - Imports the **Home** and **About** components.
- Configure routes in AppRoutes such that:
  - The root path **(/)** displays the **Home** component.
  - The **/about** path displays the **About** component.
- Create an **App** component that:
  - Wraps the application with **BrowserRouter**
  - Uses the alias name **Router** for **BrowserRouter**
- Add a navigation bar in **App.jsx** using **Link** components that:
  - Navigates to the **Home** page
  - Navigates to the **About** page

4. Using Axios to Fetch and Post Data from an API

- Install **Axios** in a React project using the appropriate **npm command.**
- Create a functional component named **DataFetchingAndPosting** that uses React hooks.
- Inside the component, use the **useState** hook to:
  - Store fetched data in a state variable named **data** (initially an empty array)
  - Store new post details in a state variable named **newPost** with the properties **title**, **body**, and **userId**
- Use the **useEffect** hook to make a **GET request** using Axios to fetch posts from the URL: *https://jsonplaceholder.typicode.com/posts* and store the response data in the data state.
- Handle errors that may occur during the GET request using the **.catch()** method.

- After a successful POST request:
    - Display the newly added post data in the console
    - Update the **data** state to include the newly created post
- Display the fetched post titles in an unordered list (**<ul>**) using the **map()** function and a unique **key**.
- Create two input fields to accept **title** and **body** values for a new post and update the **newPost** state using the **onChange** event.
- Add a button labeled "**Add Post**" that triggers the **handlePost** function when clicked.
- Import and render the **DataFetchingAndPosting** component inside **App.jsx.**

Discussion:

- How does Axios differ from fetch?
- What happens when the component mounts with useEffect?