National University of Computer and Emerging Sciences

Laboratory Manual

for

Web Engineering
(SL3003)

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

Department of Computer Science

FAST-NU, Lahore, Pakistan

Lab 8: React Components, Event Listeners, useState, Props, and Conditional Rendering

Objectives

- Understand and use state effectively with `useState`.
- Pass and manipulate props between components.
- Handle user input dynamically and update state.
- Implement controlled components with forms.
- Utilize conditional rendering in React.
- Use event listeners in React.

Instructions: Setting Up the React Environment

Step 1: Install Node.js

1. Download and install the latest LTS version of Node.js from https://nodejs.org/.
2. Verify installation by running the following commands in the terminal:
3. `node -v`
4. `npm -v`

   This should print the installed versions of Node.js and npm.

Step 2: Create a React App

1. Open your terminal and run the following command to create a new React application:
2. `npm create vite@latest my-project-name --template react`
3. Navigate into the project directory:
4. `cd my-project-name`
5. `npm install`
6. `npm run dev`
7. If you encounter any issues related to user access while running the npm install command on Windows, run the following command in PowerShell:
   `Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned -Force`

Task 1: Creating an Enhanced Movie List

Instructions:

1. Create a new React component named `MovieList` that maintains a list of movie names using `useState`.
2. Allow users to add a new movie by entering a title in an input field and clicking an "Add Movie" button.
3. Use the `Movie` component (from previous work) to display each movie in the list.
4. Each movie should have an input box that allows users to update the movie title dynamically.
5. Implement a "Remove" button to delete a movie from the list.
6. Ensure all state updates trigger a re-render.

Implementation Hints:

- Use a controlled component for the input field.
- Make sure to use a deep copy of the array when updating state.
- Use `.map()` to render the movie list dynamically.

Example Component Structure:

- `App.js`
- `MovieList.js`
- `Movie.js`

Task 2: Implementing a Simple Counter with Step Control

Instructions:

1. Create a new component named `Counter`.
2. Use `useState` to maintain a count state.
3. Add an input field where users can specify a step value (e.g., increment by 1, 2, 5, etc.).
4. Include buttons to increase and decrease the count by the step value.
5. Ensure the count never goes below zero.
6. Display the current count dynamically.

Implementation Hints:

- Use a controlled component for the step input.
- Update state properly using a function inside `setState` to ensure previous state is considered.
- Use conditional rendering to disable decrementing if `count === 0`.

Example Component Structure:

- `App.js`
- `Counter.js`

Task 3: Conditional Rendering with Toggleable Content

Instructions:

1. Create a `Profile` component that displays a user's name and description.
2. Add a "Show/Hide Details" button that toggles the description's visibility.
3. Use conditional rendering (`&&` or ternary operator) to show/hide the description.
4. Implement a theme switcher with two buttons: "Light Mode" and "Dark Mode".
5. Use state to update the background color and text color dynamically.
6. Add an event listener to the buttons to handle theme changes.

Implementation Hints:

- Use multiple `useState` hooks to handle different states.
- Use inline styles or CSS classes to update theme dynamically.
- Avoid unnecessary state updates by checking the current theme before setting a new one.

Example Component Structure:

- `App.js`
- `Profile.js`

Bonus Task (Optional): Handling Form Submission

Instructions:

1. Create a `FeedbackForm` component with input fields for name and feedback message.
2. Add a "Submit" button that logs the input values to the console.
3. Prevent form submission if any input is empty.
4. Clear input fields upon successful submission.
5. Implement event listeners for handling form submission.

Implementation Hints:

- Use event handling to capture form submission.
- Validate form inputs before processing data.
- Use `e.preventDefault();` to prevent the default form submission behavior.
- Add event listeners using `onSubmit` and `onChange` for controlled inputs.