**Portfolio (Week-7)**

This week’s lab session focused on creating functional components in React and implementing state management using Hooks. Below is a detailed explanation of the tasks performed and the lessons learned:

**Task 1: Setting Up a React Application**

To begin, I set up a new React application using the following steps:

1. **Initialize the Application**
   * Created a new directory and initialized a React project using:

A screenshot of a computer

Description automatically generated

Navigated to the project folder:

Started the development server:

A screenshot of a computer

Description automatically generated

We will now create a folder named Counter.js in the src folder:

A screenshot of a computer

Description automatically generated

1. **Create Counter Component**

* Created a new file Counter.js in the src folder.
* Imported React and other necessary hooks:

A screenshot of a computer

Description automatically generated

1. **Functional Component with State Management**

* Defined a component named Hook\_ControlledButtonState:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

This component does a few things:

* It uses useState to keep track of how many times the button is clicked.
* It has a function called ClickHandle that adds 1 to the count each time the button is clicked.
* It shows the current count and a button to click.

Now, We also need to update our index.js file to use this new component:

1. **Update index.js**

* Replaced the default <App /> component with <Hook\_ControlledButtonState />.

**Output:** A button was displayed on the webpage, and the count increased with each click.

A screenshot of a computer program

Description automatically generated

When we ran our app again, we saw a page with a button. Each time we clicked the button, the number went up by one.Making Our Emoji Counter

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Task 2: Creating an Emoji Counter**

To make the application more interactive, I developed a component to display emojis based on user interaction.

1. **Create EmojiCounter Component**
   * Created a file EmojiCounter.js in the src folder.
   * Imported React, useState, useEffect, and emoji images:

A screenshot of a computer

Description automatically generated

1. We copied three image files (Love.png, Sad.png, and Like.png) into our src folder.

A screenshot of a computer

Description automatically generated

1. At the top of our new file, we added these lines:

A screenshot of a computer

Description automatically generated

Then we made our EmojiCounter component. Here's the whole component:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

This component does a few things:

* It uses useState to keep track of which picture to show and how many times the button is clicked.
* It uses useEffect to change the picture when the props.pic value changes.
* It has a ClickHandle function that adds 1 to the count each time the button is clicked.
* It shows a button with the current count and an emoji picture.

We updated our index.js file again to use this new component:

A screenshot of a computer

Description automatically generated

We had an issue, that when we run it we had a blank screen on the page, the problem was with the spelling of the Emojicounter so when we run the program we saw two emojis, the Love emoji was not showing:

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Debugging Issues**

* Resolved a blank screen issue caused by a typo in the component name.
* Fixed an image display error by re-downloading a missing image.

A screenshot of a computer

Description automatically generated

What We LearnedIn this lab, we learned how to:

* Set up a new React project
* Make simple components that keep track of information (like how many times a button is clicked)
* Use useState to manage information in our components
* Use useEffect to do things when our component's information changes
* Show pictures in our components
* Pass information to our components using props

When we ran our app this time, we saw three buttons. Each button had a different emoji on it. When we clicked a button, the number on that button went up.

Now here are the answers to some questions from this weeks lab:

Q1. Write one page reflective what did you learn about React Hook API during this week

Ans: This week, I learned a lot about React Hook API. Here's what I discovered:React Hooks are special functions that let us use state and other React features in functional components. Before, we could only do this in class components.

The main hooks I learned about were useState and useEffect:useState lets us add state to functional components. It's really handy for keeping track of things that change, like a counter or form inputs.useEffect helps us perform side effects in our components. It's like a combination of componentDidMount, componentDidUpdate, and componentWillUnmount from class components, all in one place.I found out that hooks make our code cleaner and easier to understand. We don't need to switch between classes, higher-order components, and render props anymore.One cool thing about hooks is that we can create our own custom hooks. This lets us reuse stateful logic between different components without changing our component hierarchy.I also learned that hooks follow some rules:

* We should only call hooks at the top level of our components, not inside loops or conditions.
* We should only call hooks from React functional components or custom hooks, not regular JavaScript functions.

Overall, I feel like hooks have made React development simpler and more fun. They've given me new ways to think about and organize my code.

Q2. Study the code in EmojeeCounters.js, Please note, You Do not need to submit the full code rather you need to answer the following questions for your this week portfolio

Ans: The name of the component created in EmojeeCounters.js is EmojiCounter.

The lines of code that use the EmojiCounter in index.js are:  
<EmojiCounter pic="Love" />  
<EmojiCounter pic="Like" />  
<EmojiCounter pic="Sad" />

The states are declared in these lines:  
const [pic, setPic] = useState(Love);  
const [count, setCount] = useState(0);  
These lines create two state variables: 'pic' for the emoji image and 'count' for the number of clicks.

The event handler is associated in this line:  
<button onClick={ClickHandle}>

In this line, 'pic='Love'' is a prop being passed to the EmojiCounter component. It tells the component to display the 'Love' emoji.

useEffect is a hook that lets us perform side effects in functional components. In this component, it's used to update the 'pic' state based on the 'pic' prop. It runs whenever the 'pic' prop changes.

<p>{props.pic} <span></span> <button onClick={ClickHandle}>{count } <img src={pic} alt=""/> </button> </p> </div> ) } This code returns the JSX for the component. It creates a div with a paragraph that shows the name of the emoji (props.pic), a button that shows the click count and has the ClickHandle function attached to its onClick event, and an img tag that displays the emoji image.

Q3. Create a code for a Component that takes two HTML one text box and one label.

Label will be used to display the images. So it should be like this  
If I write "Happy" in the text box the label should show happy face (You can use any image)  
If I write "Like" in the text box the label should show Like icon  
If I write "sad " the label should show sad emoji.

Ans: Here's a simple React component that does what you've described:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

This component uses a text input and a label to display emojis based on the input. It uses useState to manage the input text and the current emoji, and useEffect to update the emoji whenever the text changes. The handleChange function updates the text state as the user types. The label displays the emoji image if one is selected based on the input.

**Key Learnings**

* **Setting up React Projects:** Using create-react-app simplifies project initialization.
* **Hooks:** Hooks like useState and useEffect provide powerful tools for managing state and side effects in functional components.
* **Debugging:** Errors such as typos and missing assets can cause issues but are solvable with careful debugging.

This week’s lab reinforced the importance of React Hooks in building dynamic and interactive applications, enhancing my confidence in using them effectively.