Rajeshwari Reddy Gummadi

U2276440

Reflection of Lab Exercise week 7

**Web and Mobile Application Development**

Task 1

Create a functional component for counting clicks. The user interface of the component would just have one button and heading. The component would record number of the clicks in the state variable of the button component and will increment as on every click of the button.

A blue screen with white text

Description automatically generated

A blue screen with white text

Description automatically generated

Task 2

Emoji Counter in this task we will add an image for three emojis Love, Like, sad emotions.

A screenshot of a computer

Description automatically generated

Every time you click on the image it shows an increment in number.

A screenshot of a computer

Description automatically generated

Portfolio for this week

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

This week, I explored the React Hook API, concentrating on utilising hooks to manage state and create functional components. I set up a new React project, made a click counter, and made an emoji counter with dynamic picture rendering. These exercises gave me a thorough grasp of how hooks improve component functionality and modernise state management in React.

The primary reason for utilising a React hook is to comprehend React Hook fundamentals, especially useState and useEffect.   
to incorporate functional components' state management.  
to generate content dynamically using props and state.

The key points I have learned are as follows:

* **Setting Up a React Project**:

I learned how to initialize a new React project using npx create-react-app, which streamlined the setup process and provided a structured environment for development.

* useState Hook:   
    
  In order to manage state within functional components, the useState hook was essential. I knew how state variables were declared and updated in reaction to user input. For example, useState was used to track the number of clicks in the click counter component.
* useEffect Hook:   
    
  In order to handle side effects and initialize component state depending on props, the useEffect hook was essential. I discovered how to make sure the component dynamically renders the proper image by using useEffect to update the state when props change.
* Event Handling:   
    
  Another important factor was the implementation of event handlers in functional components. I developed routines to deal with click events, which caused re-renders and state updates. Both the click counter and the emoji counter components made this clear.
* Dynamic rendering and props:   
    
  One important learning experience was using props to render material dynamically by passing them to components. With the help of props, the emoji counter component showed how to choose which picture to show and how to update the state appropriately.

The skills acquired this week have practical applications in building interactive and dynamic web applications. Understanding hooks allows for cleaner, more maintainable code, and enhances the ability to create responsive user interfaces.

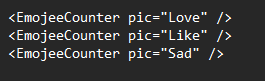
Q2. Study the code in EmojeeCounters.js, Please note, You Do not need to submit the full code rather you need to answer the following questions for your this week portfolio

• What is Name of the Component you have created in EmojeeCounters.js

Answer: EmojeeCounter

* Identify the line of code that uses the EmojeeCounter in index.js

Answer:



* Declares the states of each of the html elements defined in the EmojeeCounters.js ( identify these lines and explain only those lines )

Answer:



State for the Image (pic): A state variable called pic and a function called setPic to update it are declared in this line. The image of Love is established as the first state. The picture that is currently displayed is stored in this state variable, which is updated in response to the pic prop that is supplied to the component.



State for the Click Count (count): A state variable count and a function setCount to update it are declared in this line. Zero is the initial state. The number of times the image has been clicked is tracked by this state variable.

* Explain the line : , what is pic=’Love’ means in this line.

Answer:

The EmojeeCounter component is rendered using the JSX syntax .   
pic="Love": The EmojeeCounter component is receiving this prop. Here, "Love" is the value being affixed to the prop, which is called "pic."

what is pic=’Love’ means:

* The EmojeeCounter component uses the pic prop to decide which picture to show.
* The text value "Love" specifies which image ought to be shown. This value is used to set the image's state inside the EmojeeCounter component.

The picture prop with the value "Love" is sent to the EmojeeCounter component, which utilizes it to decide which image to show. This is how it takes place:

* Receiving the Prop:

A screenshot of a computer program

Description automatically generated

* **Using useEffect to Update the Image**:

A screen shot of a computer program

Description automatically generated

The component in this useEffect hook verifies props.pic's value. If "Love" is selected, the Love image is used as the picture state. This guarantees that, depending on the prop value supplied to the component, the appropriate picture is shown.   
Thus, pic="Love" indicates that the Love emoji image will be displayed by the EmojeeCounter component.

* What is useEffect and why you think we have used it in the Component.

Answer:

React's useEffect hook enables you to implement side effects in your functional components. Operations that have an impact on something outside the purview of the function being performed are known as side effects. Examples of this include data retrieval, subscriptions, and manual DOM modifications.

A black screen with white text

Description automatically generated

* Callback Function: Your side effect code is placed in the function that is the first argument.
* Dependents Array: An array of dependents is the second input. Only if one of the dependencies has changed will the effect be rerun.

A computer screen with white text

Description automatically generated

Using the pic prop that is supplied to the EmojeeCounter component, useEffect is utilized to refresh the image's (pic) state.   
Initial Setup: useEffect makes sure that the right picture is shown based on the initial pic parameter value when the component is first rendered.   
Reacting to Prop Changes: useEffect will rerun and update the state to reflect the new prop value if the picture prop changes while the component is mounted. This guarantees that the component will always show the right image.

To keep the user interface consistent with the component's props, the EmojeeCounter component uses useEffect to dynamically alter the displayed picture if the pic prop changes.

* Explain these line of the codes in functional component EmojeeCounter.js:

return (

<div className="App">

<p>{props.pic} <span></span>

<button onClick={ClickHandle}>{count }

<img src={pic} alt=""/>

</button>

</p>

</div>

)

}

Answer:

* The return statement constructs the JSX to be rendered by the EmojeeCounter component.
* It includes a div container with a class name for styling.
* A paragraph element displays the name of the emoji.
* A button element displays the click count and contains an image. The button increments the count when clicked.
* The image source is dynamically set based on the component's state.

 This div element has the class name "App" assigned to it. Certain CSS styles can be applied to this container using the class name.

 This element is a paragraph.   
{props.pic}: The value of the picture prop is inserted into the paragraph using this phrase. It shows the emoji's name, such as "Love," "Like," or "Sad."   
: This span element is empty. Although it has no characteristics or content in this context, it can be used for layout or styling.

 : This element is a button.   
onClick={ClickHandle}: This assigns the ClickHandle function to the onClick event handler. The ClickHandle function is triggered when the button is pressed, increasing the count state.

 The current value of the count state is inserted into the button via this expression. It shows how many times you've clicked the button.

 : This element is an image.   
src={pic}: The URL of the image to be displayed is stored in the pic state variable, which is set to the value of the src attribute. The props are used to dynamically update this state.picture value.   
alt="": Alternative text for the image is provided via the alt element. Since it's an empty string in this instance, no other text is offered.

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. Run this component take the screen shot of your newly run component and write a paragraph how did you develop this component.

Answer:

* Depending on what is typed in the text box, the EmojiDisplay component dynamically modifies the image that is shown.
* It manages the text and graphics states using useState.
* Every time the text state changes, the useEffect hook modifies the image state.
* As the user types, the handleChange function modifies the text state.
* Depending on the current state, the return statement renders the input box and the picture.

Code index.js

A screen shot of a computer program

Description automatically generated

EmojiDisplay

A screen shot of a computer screen

Description automatically generated A black background with colorful lines

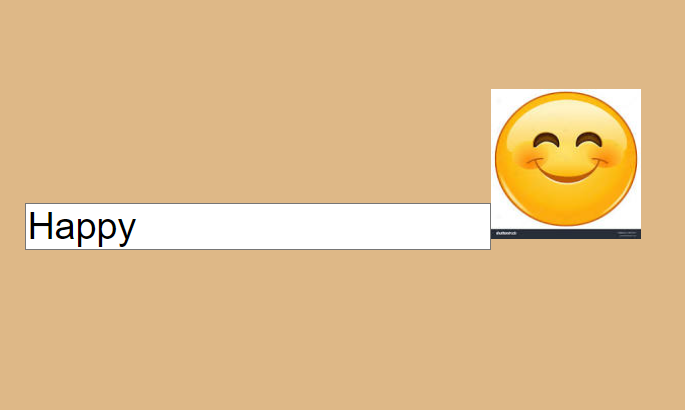
Description automatically generated

A close up of a text

Description automatically generated

This is how the page looks like before and after inserting into the textbox.

When you insert Happy it shows happy emoji.



A blue button on a brown background

Description automatically generated

When Like text is inserted you get Like icon

When Sad text is inserted, you get sad emoji

A yellow and white emoji with a white stripe

Description automatically generated with medium confidence