Week 7 Portfolio

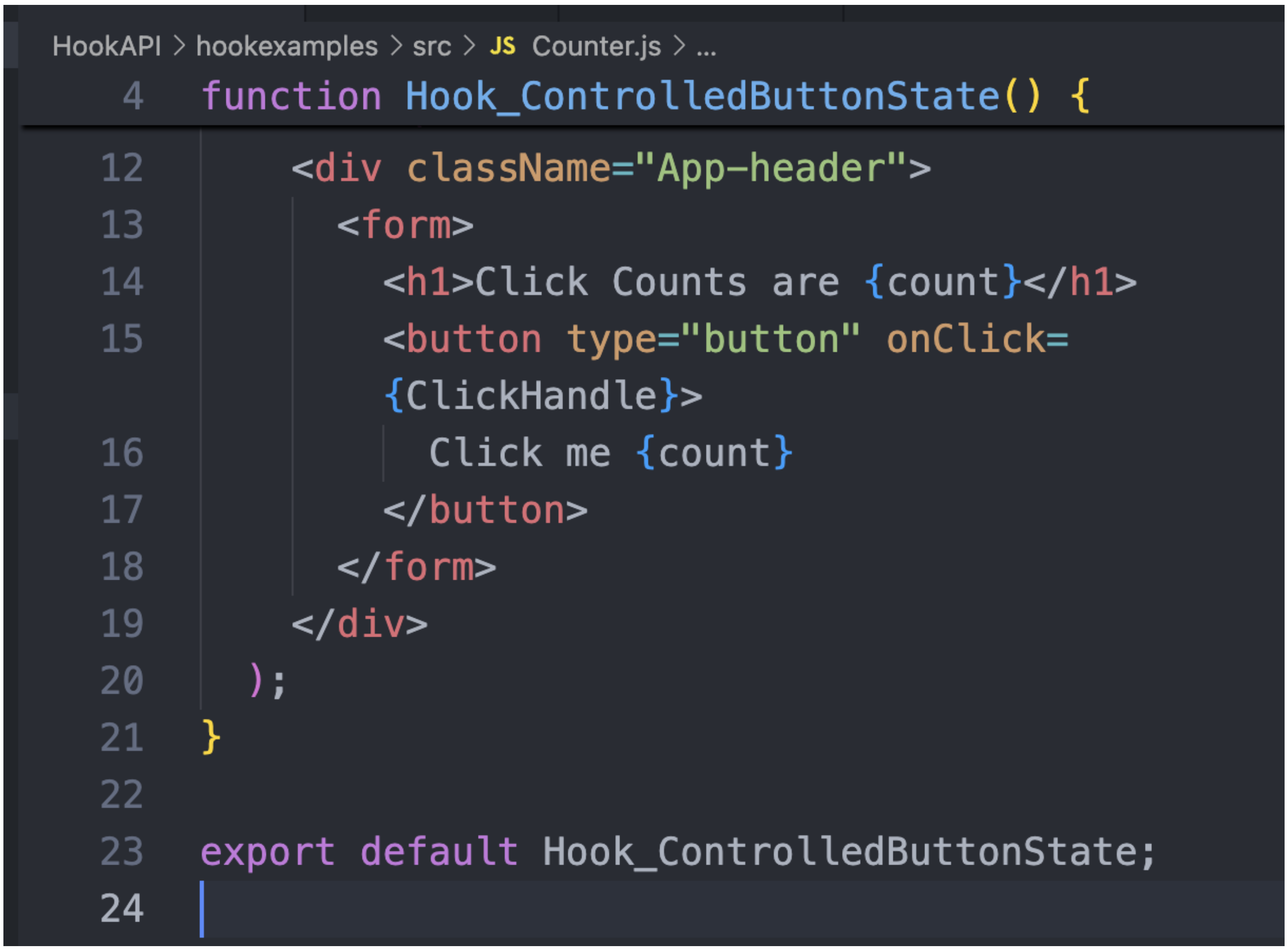
Name: Deepak Kumar
Student ID: 2284279

React Hook API Tutorial

1. Screenshots

Task 1 Create a Click Counter Component

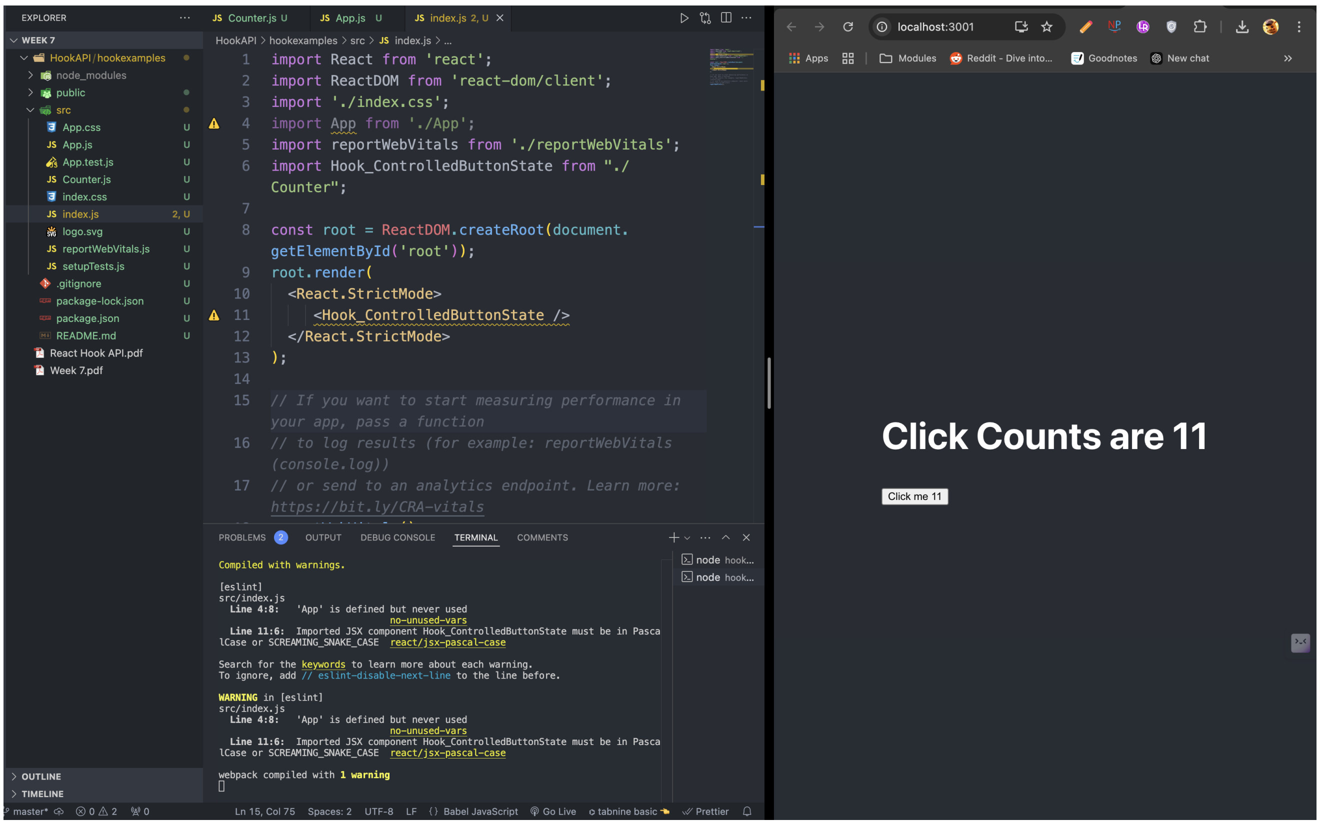
First, I created a React app using the `npx create-react-app hookexamples` command to set up the project. Once the app was ready, I made a new file called `Counter.js` in the `src` folder to implement the click counter functionality.



HookAPI > hookexamples > src > JS Counter.js > ...

```
function Hook_ControlledButtonState() {
    <div className="App-header">
      <form>
        <h1>Click Counts are {count}</h1>
        <button type="button" onClick=
        {ClickHandle}>
          Click me {count}
        </button>
      </form>
    </div>
  );
}

export default Hook_ControlledButtonState;
```

In this file, I used the `useState` hook to keep track of how many times the button was clicked. Then, I updated the `index.js` file to render the `Counter` component on the browser.



```javascript
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import Hook_ControlledButtonState from "./Counter";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Hook_ControlledButtonState />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
```

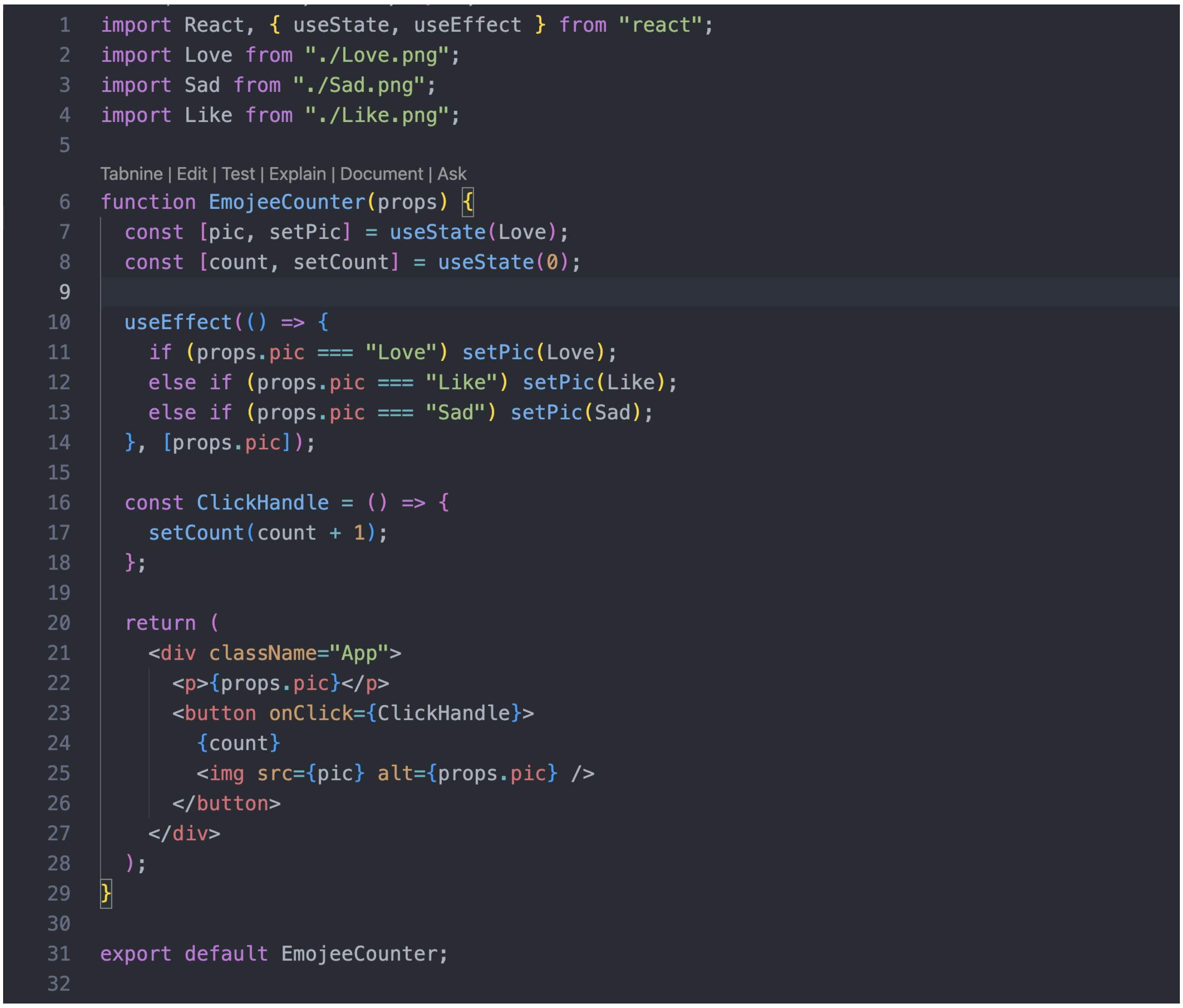
```
Compiled with warnings.

[eslint]
src/index.js
  Line 4:8:   'App' is defined but never used
                             no-unused-vars
  Line 11:6:  Imported JSX component Hook_ControlledButtonState must be in Pasca
lCase or SCREAMING_SNAKE_CASE  react/jsx-pascal-case

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

WARNING in [eslint]
src/index.js
  Line 4:8:   'App' is defined but never used
                             no-unused-vars
  Line 11:6:  Imported JSX component Hook_ControlledButtonState must be in Pasca
lCase or SCREAMING_SNAKE_CASE  react/jsx-pascal-case

webpack compiled with 1 warning
```

Click Counts are 11

Task 2 - Create Emoji Counter Component

For this task, I added the images `Love.png`, `Sad.png`, and `Like.png` to the `src` folder of my project.

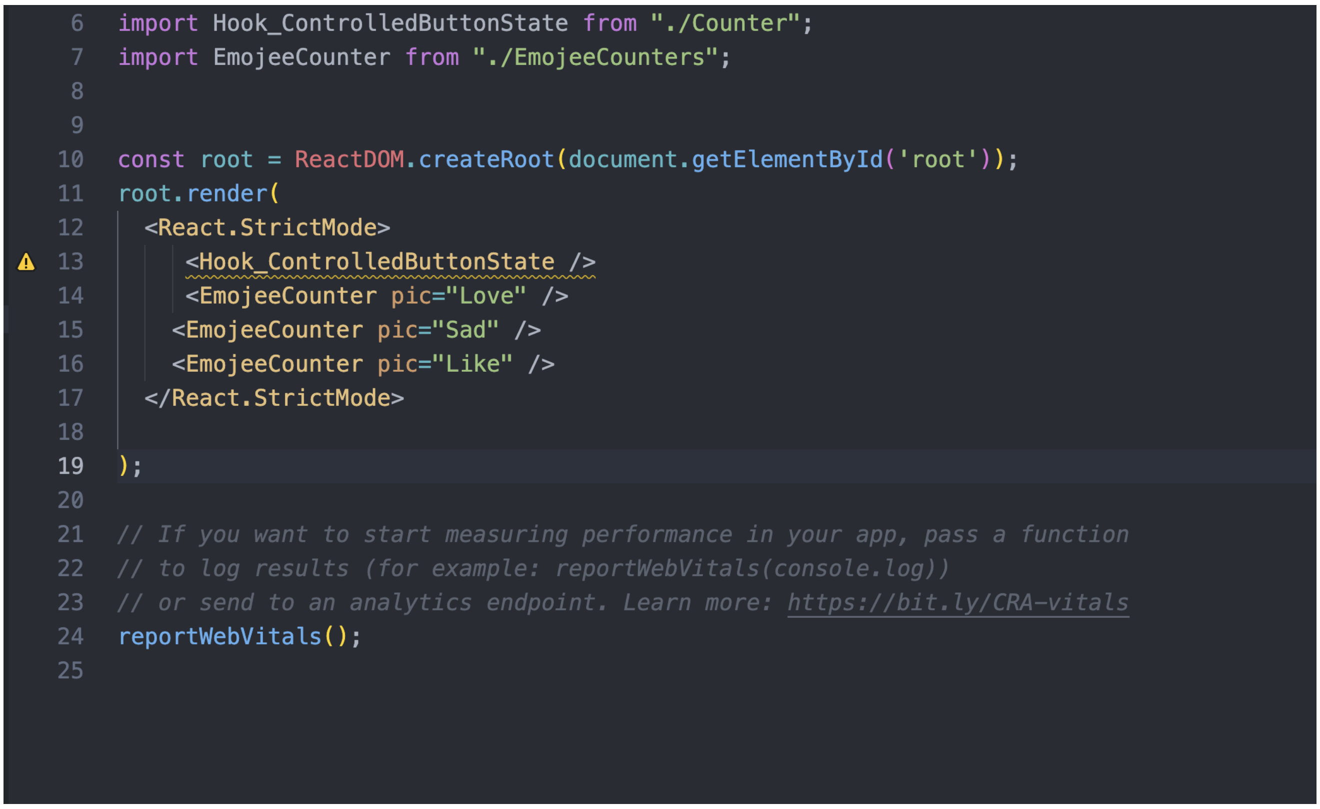
After that, I created a new file called `EmojeeCounters.js` to build a component that displays an emoji and tracks its click count.



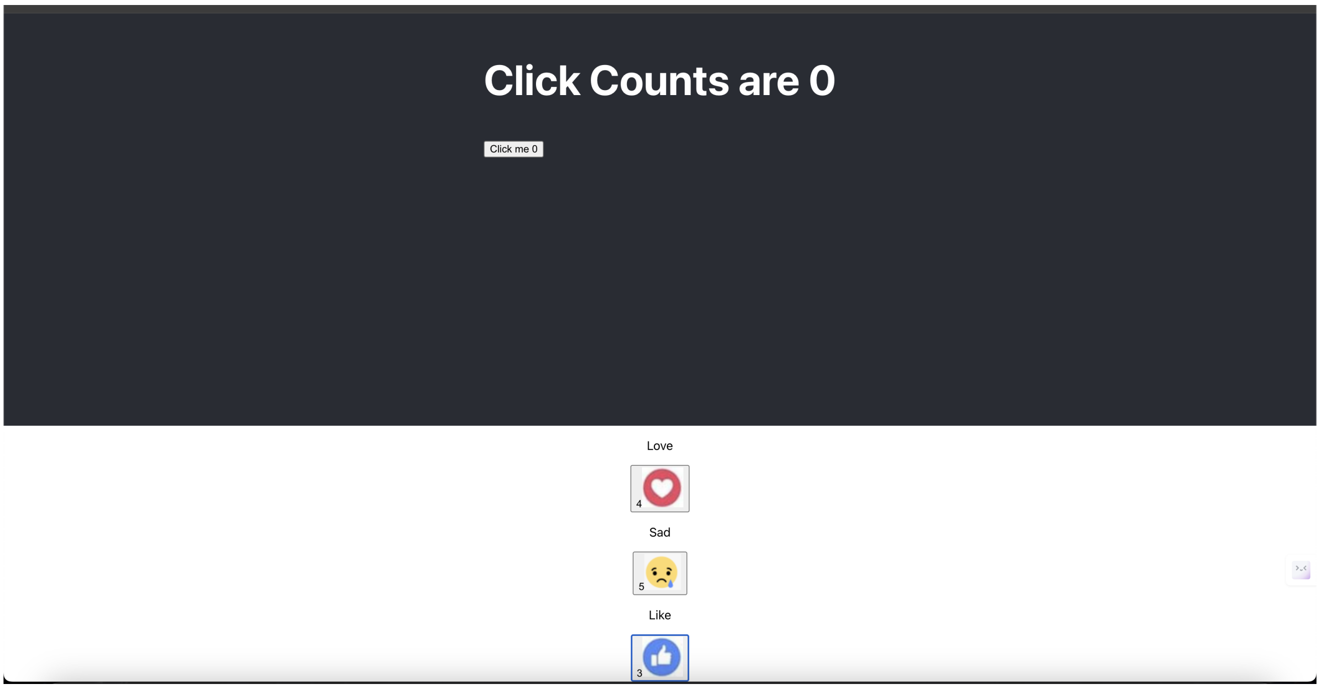
```
import React, { useState, useEffect } from "react";
import Love from "./Love.png";
import Sad from "./Sad.png";
import Like from "./Like.png";

function EmojeeCounter(props) {
  const [pic, setPic] = useState(Love);
  const [count, setCount] = useState(0);

  useEffect(() => {
    if (props.pic === "Love") setPic(Love);
    else if (props.pic === "Like") setPic(Like);
    else if (props.pic === "Sad") setPic(Sad);
  }, [props.pic]);

  const ClickHandle = () => {
    setCount(count + 1);
  };

  return (
    <div className="App">
      <p>{props.pic}</p>
      <button onClick={ClickHandle}>
        {count}
        <img src={pic} alt={props.pic} />
      </button>
    </div>
  );
}

export default EmojeeCounter;
```

I used the `useState` hook to handle the click counter and the `useEffect` hook to dynamically update the image based on the `props.pic` passed to the component.

Then, I updated `index.js` to include three `EmojeeCounter` components, each with a different emoji.



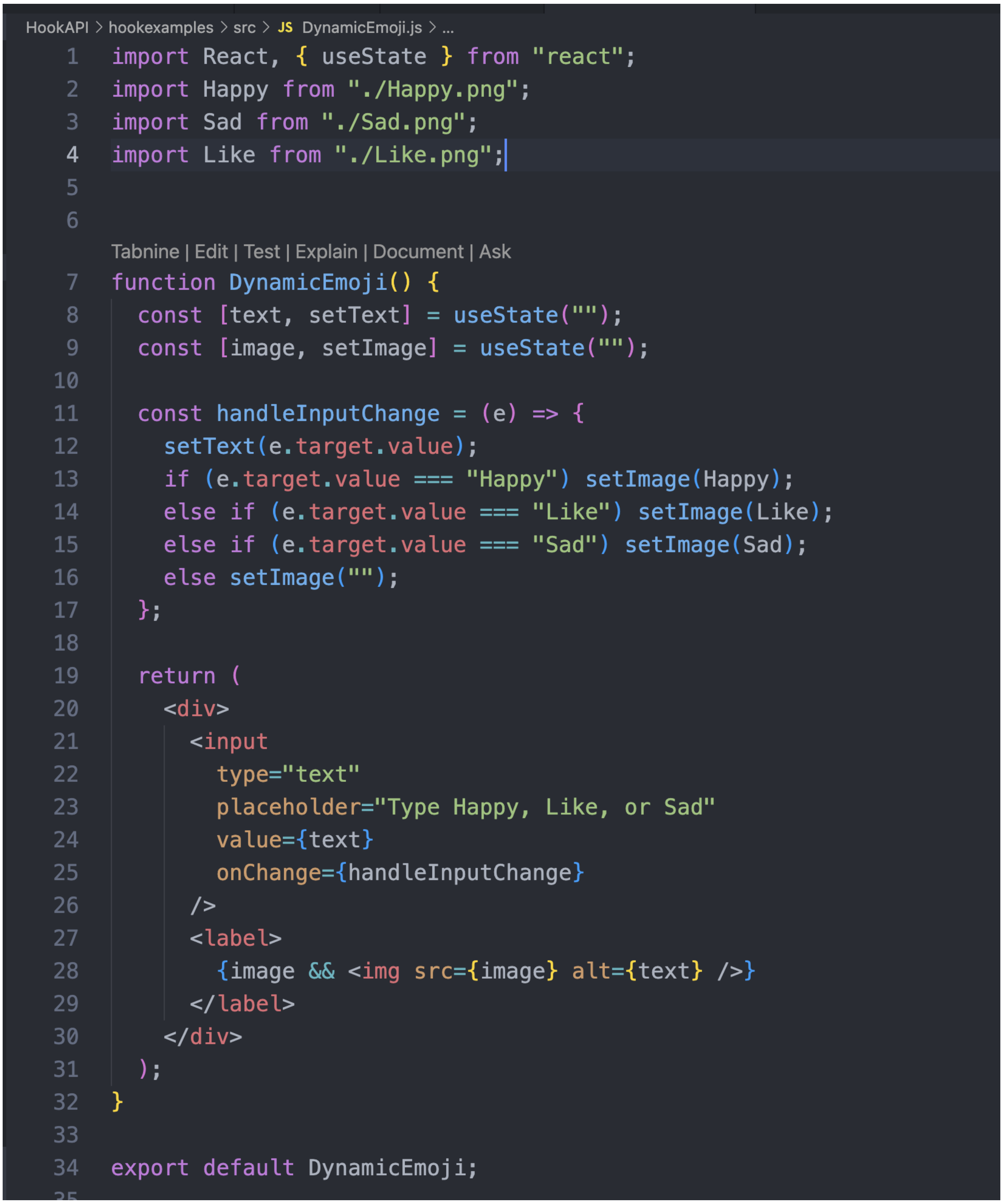
```
import Hook_ControlledButtonState from "./Counter";
import EmojeeCounter from "./EmojeeCounters";


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
     <Hook_ControlledButtonState />
     <EmojeeCounter pic="Love" />
    <EmojeeCounter pic="Sad" />
    <EmojeeCounter pic="Like" />
  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```



```
Click Counts are 0

Click me 0

Love
4
Sad
5
Like
3
```

Task 3 - Create Text Box and Label Component

To create this component, I started by making a new file called `DynamicEmoji.js` and adding it to the `src` folder. I imported the `Happy.png`, `Sad.png`, and `Like.png` images to use in the component.

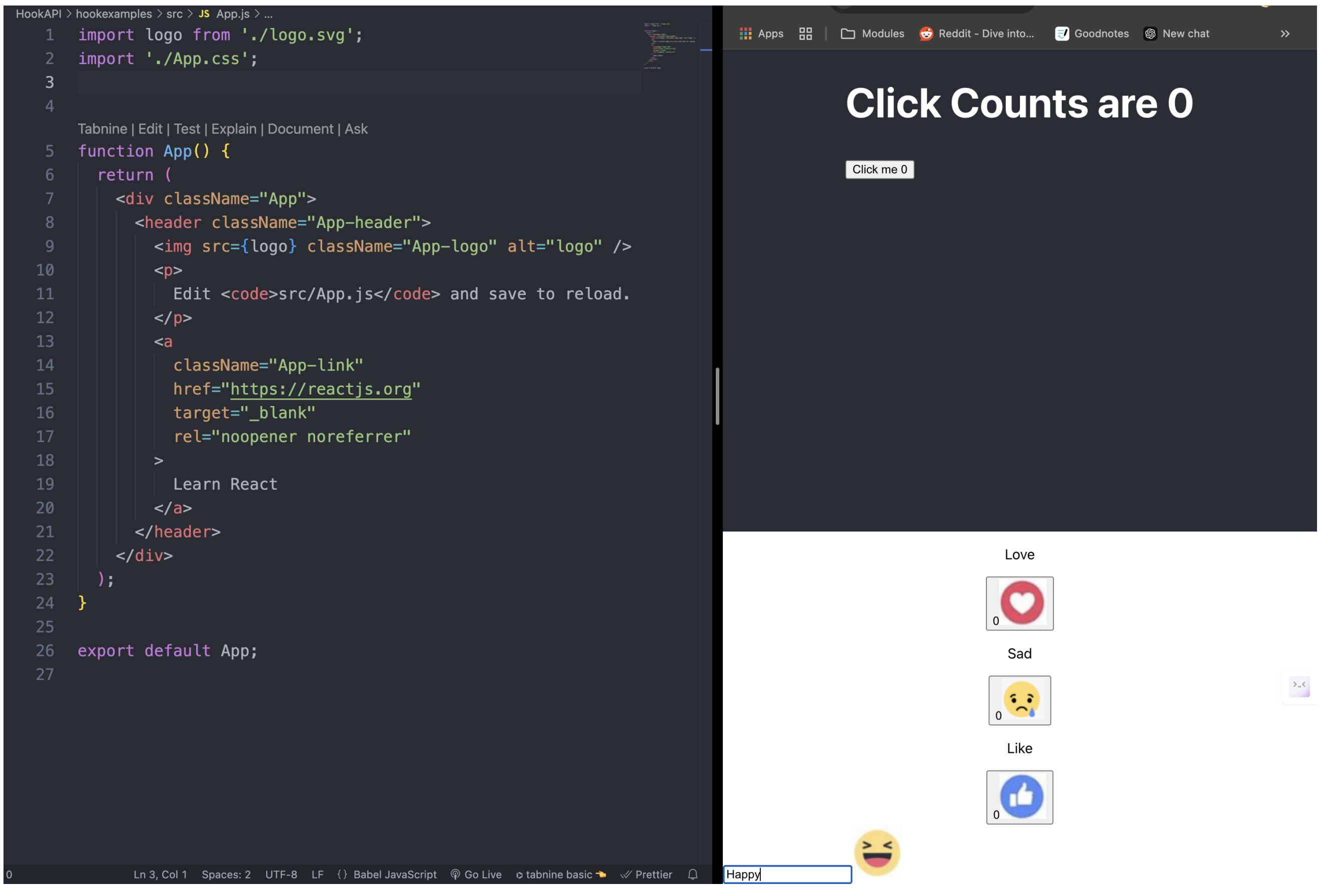


HookAPI > hookexamples > src > JS DynamicEmoji.js > ...
Tabnine | Edit | Test | Explain | Document | Ask

```
import React, { useState } from "react";
import Happy from "./Happy.png";
import Sad from "./Sad.png";
import Like from "./Like.png";


function DynamicEmoji() {
  const [text, setText] = useState("");
  const [image, setImage] = useState("");

  const handleInputChange = (e) => {
    setText(e.target.value);
    if (e.target.value === "Happy") setImage(Happy);
    else if (e.target.value === "Like") setImage(Like);
    else if (e.target.value === "Sad") setImage(Sad);
    else setImage("");
  };

  return (
    <div>
      <input
        type="text"
        placeholder="Type Happy, Like, or Sad"
        value={text}
        onChange={handleInputChange}
      />
      <label>
        {image && <img src={image} alt={text} />}
      </label>
    </div>
  );
}

export default DynamicEmoji;
```

Then, I used the `useState` hook to track the user's input in a text box and dynamically update the displayed image based on the text entered.

At last, I updated `index.js` to render the `DynamicEmoji` component, ran the app, and tested the functionality by typing different words to see the matching emoji.


HookAPI > hookexamples > src > JS App.js > ...

```
import logo from './logo.svg';
import './App.css';


Tabnine | Edit | Test | Explain | Document | Ask
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

Ln 3, Col 1 Spaces: 2 UTF-8 LF {} Babel JavaScript Go Live tabnine basic Prettier

Click Counts are 0

Click me 0

Love

0

Sad

0

Like

0

Happy

What you need to write for your Portfolio of this Week

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

Working on this lab was fun and a bit frustrating at the same time. Setting up the React project was pretty straightforward since I've done it before (in week 5), so that part felt easy, and it gave me a little confidence. But then, writing the click counter took me some time because I kept forgetting to import `useState`. Once I figured it out with the help of my group mates, seeing the counter actually working when I clicked the button felt really cool.

The emoji counter was a bit trickier. Adding the images to the project seemed easy at first, but I messed up the file paths a couple of times, and the images didn't show up. Fixing that taught me to double-check the file locations every time. I also learned how `useEffect` works, which was interesting but slightly confusing at first because I didn't realize why it was needed. After experimenting a bit, it made sense that it helps update things dynamically when props change.

The dynamic emoji task was the most creative part. I liked how typing in the text box instantly updated the image. The hardest thing was making the logic for showing the right emoji, but I eventually got it (with the help of chatgpt).

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 the name of the component you have created in EmojeeCounters.js?**
  `EmojeeCounter`.
- **Identify the line of code that uses the EmojeeCounter in index.js.**
  `<EmojeeCounter pic='Love' />`.
- **Declare the states of each of the HTML elements defined in EmojeeCounters.js (identify and explain those lines).**
  `const [pic, setPic] = useState(Love);` this code manages the image, and this `const [count, setCount] = useState(0);` tracks the clicks.
- **What lines of code associate the event handler used?**
  `<button onClick={ClickHandle}>` connects the button click to the `ClickHandle` function.
- **Explain the line `<EmojeeCounter pic='Love' />`. What does `pic='Love'` mean?**
  It passes the string `Love` as a prop to the component to display the love emoji.
- **What is useEffect, and why have we used it in the component?**
  `useEffect` updates the image dynamically based on the `pic` prop whenever it changes.
- **Explain the return statement in EmojeeCounters.js.**
  It renders a `<div>` with the emoji name, a button showing the count, and the emoji image we pass.