Lesson 03 Demo 05
Create a New React Component Using useState Hook

Objective: To create a React component demonstrating the use of the **useState** hook

Tools Required: Node terminal, React app and Visual Studio Code

Prerequisites: Knowledge of creating a React app and understanding of the folder structure
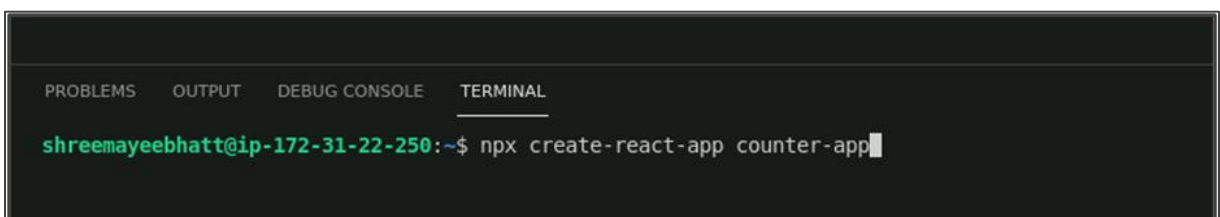

Steps to be followed:

1. Create a new **React** app
2. Configure the file **src/App.js**
3. Create a new file called **Counter.js**
4. Run the app and verify the functionality

Step 1: Create a new React app

1.1 Open your terminal and run the following command to create a new React app:
   npx create-react-app counter-app

This will create a new **React** app in a directory named **counter-app.**

1.2 Navigate into the newly created project directory by running the following command: **cd counter-app**
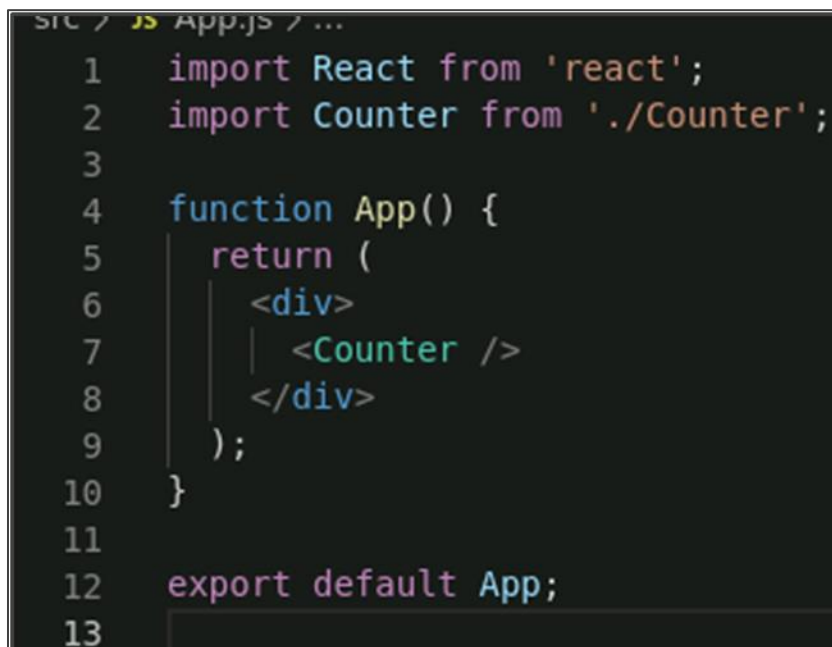
Step 2: Configure the file src/App.js

2.1 Open the React project in the Visual Studio code and navigate through the project
directory of **counter-app** to open the **src/App.js** file. and replace existing code in **App.js**
with the following code:

import React from 'react';
 import Counter from './Counter';

function App() {
 return (
  <div>
   <Counter />
  </div>
 );
}

export default App;

```
src > JS App.js > ...
  1    import React from 'react';
  2    import Counter from './Counter';
  3
  4    function App() {
  5      return (
  6        <div>
  7          <Counter />
  8        </div>
  9      );
 10    }
 11
 12    export default App;
 13
```

This code defines the **App** component that renders the **Counter** component.

Step 3: Create a new file called Counter.js

3.1 In your code editor, create a new file called **Counter.js** inside the **src** directory
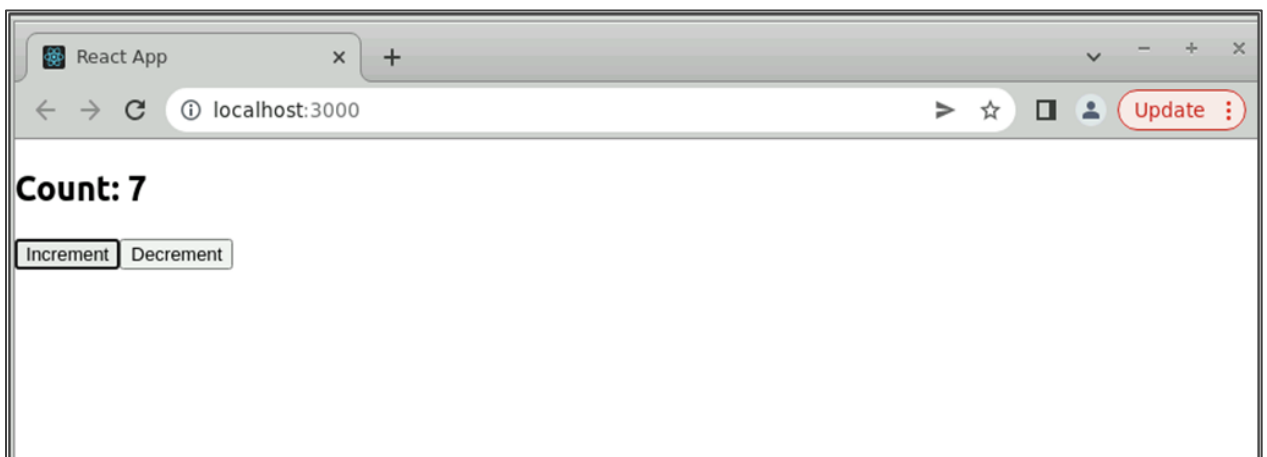
3.2 Copy and paste the provided code into the Counter.js file:

```
import React, { useState } from 'react';
const Counter = () => {
  const [count, setCount] = useState(0);
  const handleIncrement = () => {
    setCount(count + 1);
  };
  const handleDecrement = () => {
    if (count > 0) {
      setCount(count - 1);
    }
  };
  return (
    <div>
      <h2>Count: {count}</h2>
      <button onClick={handleIncrement}>Increment</button>
      <button onClick={handleDecrement}>Decrement</button>
    </div>
  );
};
export default Counter;
```

```
JS App.js        JS Counter.js ×
src > JS Counter.js
   1    import React, { useState } from 'react';
   2
   3    const Counter = () => {
   4      const [count, setCount] = useState(0);
   5
   6      const handleIncrement = () => {
   7        setCount(count + 1);
   8      };
   9
  10      const handleDecrement = () => {
  11        if (count > 0) {
  12          setCount(count - 1);
  13        }
  14      };
  15
  16      return (
  17        <div>
  18          <h2>Count: {count}</h2>
  19          <button onClick={handleIncrement}>Increment</button>
  20          <button onClick={handleDecrement}>Decrement</button>
  21        </div>
  22      );
  23    };
  24
  25    export default Counter;
  26    |
```

Step 4: Run the app and verify the functionality

4.1 Go to the terminal and execute the command **npm start** within the project directory **counter-app** to run the app

4.2 Once the server starts successfully, open **http://localhost:3000** in your browser to view the app

If the count is more than 0, you will observe that the Counter component has represented the initial count value as 0. To increase the count value by 1 and lower the count value by 1, click the **Increment and Decrement buttons**, respectively.

With this, you have successfully created a React component where buttons trigger the corresponding event handlers to update the count state. Thus, demonstrating the use of the **useState** hook.