COMP 9783 Front-end Development

Lab-4-3 - React useState Lab. (4% of the course mark)

Name:

Student Number:

In this lab, participants will learn how to use the useState hook in React to manage state in functional components. The lab will cover the basics of state management, updating state, and ensuring the UI reflects state changes. By working through practical exercises, participants will gain hands-on experience with common use cases of useState, such as form handling, toggling UI elements, and managing complex state objects.

Lab objectives:

1. Understand the concept of state in React components.
2. Learn how to declare state variables using the useState hook.
3. Practice updating state and ensuring the UI re-renders correctly.
4. Explore common patterns and best practices for using the useState hook.

Create a react app:

On **VSCode**, open the **terminal** and type the following commands:

1. **create-react-app react-use-state-app** and **press enter**. This will create the files and folders required by React.
2. On the **src folder**, **delete** the following files:
    a. App.css
    b. App.test.js
    c. index.css
    d. logo.svg
    e. reportWebVitals.js

    f. setupTests.js
3. **Modify ./src/index.js** and **remove the following code** and **save the changes**:
    a. import './index.css';
    b. import reportWebVitals from './reportWebVitals';
    c. // If you want to start measuring performance in your app, pass a function
    d. // to log results (for example: reportWebVitals(console.log))
    e. // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
    f. reportWebVitals();
4. **Modify ./public/index.html** and **update the title html code** and **save the changes**:
    a. <title>react-use-state-app</title>
5. Inside **src** create a folder named: **components**, we will use this to organize **function components**.
6. This lab will use **tailwind** as the **css frontend**. To install **tailwind** type: **npm install -D tailwindcss** and **press enter**.
7. **Initialize tailwind config**, type: **npx tailwindcss init** and **press enter**.
8. Edit **tailwind.config.js**, look for the **content** attribute, add the following lines below and **save the changes**.

content: ["./src/**/*.{html,js}"]

9. Create a file named **TW.css** with the values below and **save the changes**:

@tailwind base;
@tailwind components;
@tailwind utilities;

10. Test if the tailwind setup works by doing the following:
    a. Overwrite **App.js** with the following code and **save the changes**.

```
import "./TW.css";

function App() {
  return (
    <h1 class="text-4xl font-bold text-blue-500 text-center my-4">
      Hello, World!
    </h1>
  );
}

export default App;
```

    b. Run the app and type: **npm run start** and **press enter**. Ensure that your output is the **same** as the **image below**, otherwise review the previous steps before proceeding.

Using the useState hook V1:

1. Inside the **./src/components folder**, create a file named: **UseStateV1.js**.
2. **Overwrite UseStateV1.js** with the following codes below and **save the changes**:

```
import { useState } from "react";

function UseState() {
  const [firstNameValue, setFirstNameValue] = useState("");
  const [lastNameValue, setLastNameValue] = useState("");
```

```
const fullName = `${firstNameValue} ${lastNameValue}`;

const submitHandler = (event) => {
  event.preventDefault();
  alert(
    `Form Values:\nFirst name: ${firstNameValue}\nLast name: ${lastNameValue}`
  );
};

const changeHandler = (event) => {
  if (event.target.id === "firstNameInput") {
    setFirstNameValue(event.target.value);
  } else if (event.target.id === "lastNameInput") {
    setLastNameValue(event.target.value);
  }
};

return (
  <div className="container my-0 mx-auto p-4">
    <div className="bg-gray-100 p-4 rounded shadow-md w-full">
      <h1 className="text-center text-4xl font-bold">Use State</h1>
    </div>
    <div className="bg-gray-100 p-4 rounded shadow-md w-full mt-4">
      <h2 className="text-center text-3xl font-bold">
        Service Later Ticket System
      </h2>
      <form onSubmit={submitHandler}>
        <div className="mb-4">
          <label
```

```
          htmlFor="input"
          className="block text-gray-700 text-lg font-bold mb-2"
        >
          First name:
        </label>
        <input
          type="text"
          id="firstNameInput"
          value={firstNameValue}
          onChange={changeHandler}
          className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700
leading-tight focus:outline-none focus:shadow-outline"
        />
      </div>
      <div className="mb-4">
        <label
          htmlFor="input"
          className="block text-gray-700 text-lg font-bold mb-2"
        >
          Last Name:
        </label>
        <input
          type="text"
          id="lastNameInput"
          value={lastNameValue}
          onChange={changeHandler}
          className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700
leading-tight focus:outline-none focus:shadow-outline"
        />
```

```
        </div>
        {firstNameValue.length > 0 && lastNameValue.length > 0 && (
          <div className="flex items-center justify-between">
            <button
              type="submit"
              className="bg-blue-500 hover:bg-blue-700 text-white text-lg font-bold py-2 px-4
rounded focus:outline-none focus:shadow-outline"
            >
              Assign service ticket to: {fullName}
            </button>
          </div>
        )}
      </form>
    </div>
  </div>
 );
}

export default UseState;
```

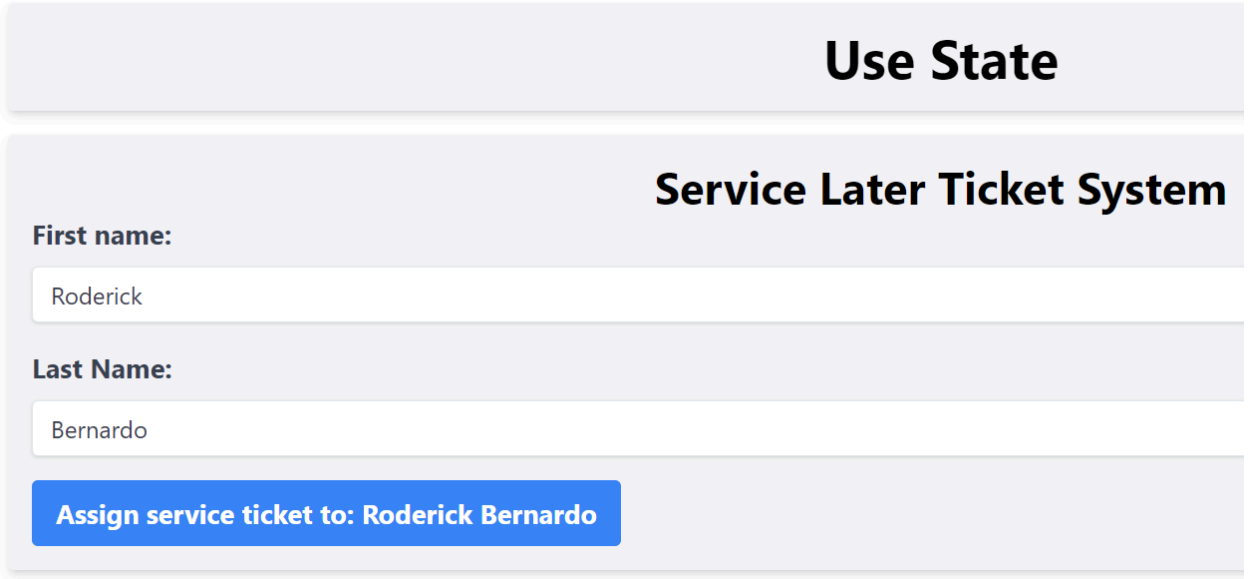
3. **Overwrite App.js** with the following code below and **save changes**.

```
import "./TW.css";
import UseStateV1 from "./components/UseStateV1";

function App() {
  return <UseStateV1></UseStateV1>;
}

export default App;
```

4. Make sure that the app is still running, if it is not running type: **npm run start** and **press enter**. Look at the browser and ensure that your output is similar below.



5. Look for the text: **Service Later Ticket System**, and underneath the closing **h2 tag**, **create another h2 tag** but **set text** to: **This ticket is for lastname, firstname**. Use **{}** and the state variables to set the values of lastname and firstname. Take a screenshot of this and name it: **useState1.png**.

Using the useState hook V2:

1. Inside the **./src/components folder**, create a file named: **UseStateV2.js**.
2. **Overwrite UseStateV2.js** with the following codes below and **save the changes**:

```
import { useState } from "react";

function UseState() {
 const [formValues, setFormValues] = useState({
  firstName: "",
  lastName: "",
 });
 const fullName = `${formValues.firstName} ${formValues.lastName}`;

 const submitHandler = (event) => {
  event.preventDefault();
  alert(
   `Form Values:\nFirst name: ${formValues.firstName}\nLast name: ${formValues.lastName}`
  );
 };

 const changeHandler = (event) => {
  const { name, value } = event.target;
  setFormValues({ ...formValues, [name]: value });
 };

 return (
  <div className="container my-0 mx-auto p-4">
   <div className="bg-gray-100 p-4 rounded shadow-md w-full">
    <h1 className="text-center text-4xl font-bold">Use State</h1>
```

```
    </div>
    <div className="bg-gray-100 p-4 rounded shadow-md w-full mt-4">
     <h2 className="text-center text-3xl font-bold">
      Service Later Ticket System
     </h2>
     <form onSubmit={submitHandler}>
      <div className="mb-4">
       <label
        htmlFor="input"
        className="block text-gray-700 text-lg font-bold mb-2"
       >
        First name:
       </label>
       <input
        type="text"
        id="firstNameInput"
        name="firstName"
        value={formValues.firstName}
        onChange={changeHandler}
        className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700
leading-tight focus:outline-none focus:shadow-outline"
       />
      </div>
      <div className="mb-4">
       <label
        htmlFor="input"
        className="block text-gray-700 text-lg font-bold mb-2"
       >
        Last Name:
```

```
          </label>
          <input
            type="text"
            id="lastNameInput"
            name="lastName"
            value={formValues.lastName}
            onChange={changeHandler}
            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700
leading-tight focus:outline-none focus:shadow-outline"
          />
        </div>
        {formValues.firstName.length > 0 &&
          formValues.lastName.length > 0 && (
            <div className="flex items-center justify-between">
              <button
                type="submit"
                className="bg-blue-500 hover:bg-blue-700 text-white text-lg font-bold py-2 px-4
rounded focus:outline-none focus:shadow-outline"
              >
                Assign service ticket to: {fullName}
              </button>
            </div>
          )}
      </form>
    </div>
  </div>
 );
}
```

```
export default UseState;
```

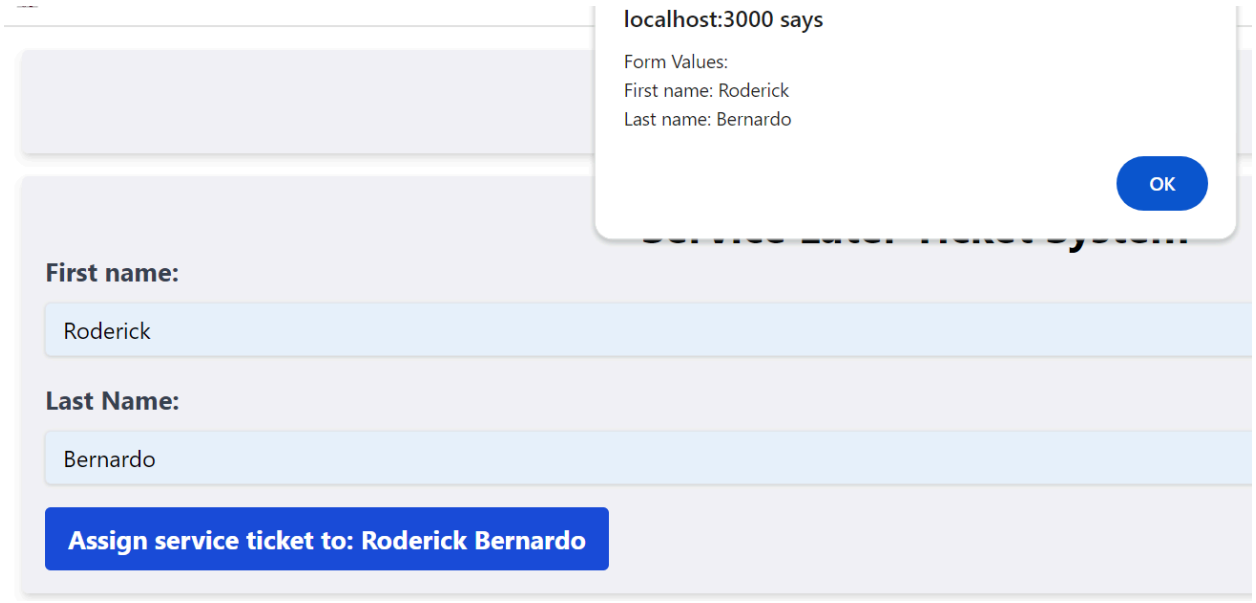
3. **Overwrite App.js** with the following code below and **save changes**.

```
import "./TW.css";
import UseStateV2 from "./components/UseStateV2";

function App() {
  return <UseStateV2></UseStateV2>;
}

export default App;
```

4. Make sure that the app is still running, if it is not running type: **npm run start** and **press enter**. Look at the browser and ensure that your output is similar below.



5. **Add another field** named for an **email address**, For reference look at the **code** used for either the **first name** or **last name**. Also **modify** the **alert code** to **show** the **new email address field**. Take a screenshot of this and name it: **useState2.png**.

Using the useState hook V4:

1. We are using **Lucide** as our **icon library**, before installing ensure that you are in the **root folder of the app** then type: **npm install lucide-react** and **press enter**.
2. Inside the **./src/components folder**, create a file named: **UseStateV4.js**.
3. **Overwrite UseStateV4.js** with the following codes below and **save the changes**:

```
import { useState } from "react";
import { Atom } from "lucide-react";

const sizeValues = [16, 32, 64, 128];
const strokeWidthValues = [1, 2, 3, 4, 5];

function UseStateV4() {
  const [size, setSize] = useState(16);
  const [color, setColor] = useState("#000");
  const [strokeWidth, setStrokeWidth] = useState(1);

  const sizeChangeHandler = (event) => {
    setSize(event.target.value);
  };

  const colorChangeHandler = (event) => {
    setColor(event.target.value);
  };
```

```
 const strokeWidthChangeHandler = (event) => {
  setStrokeWidth(event.target.value);
 };
 return (
  <div className="container my-0 mx-auto p-4">
   <div className="bg-gray-100 p-4 rounded shadow-md w-full">
    <h1 className="text-center text-4xl font-bold">
     Change the Lucide icon properties
    </h1>
   </div>

   <div className="flex flex-row justify-center bg-gray-100 p-4 rounded shadow-md w-full
mt-4">
    <div>
     <Atom size={size} color={color} strokeWidth={strokeWidth}></Atom>
    </div>
   </div>
   <div className="bg-gray-100 p-4 rounded shadow-md w-full mt-4">
    <div className="mb-4">
     <label
      htmlFor=""
      className="block text-gray-700 text-lg font-bold mb-2"
     >
      Size:
     </label>

     <select
      className="shadow border rounded w-full py-2 px-3 text-gray-700 leading-tight
focus:outline-none focus:shadow-outline"
```

```
          onChange={sizeChangeHandler}
        >
          {sizeValues.map((sizeValue) => (
            <option key={sizeValue}>{sizeValue}</option>
          ))}
        </select>
      </div>
      <div className="mb-4">
        <label
          htmlFor=""
          className="block text-gray-700 text-lg font-bold mb-2"
        >
          Color:
        </label>
        <input
          className="shadow border rounded w-full leading-tight focus:outline-none
focus:shadow-outline"
          type="color"
          onChange={colorChangeHandler}
        />
      </div>
      <div className="mb-4">
        <label
          htmlFor=""
          className="block text-gray-700 text-lg font-bold mb-2"
        >
          Stroke Width:
        </label>
```

```
        <select
          className="shadow border rounded w-full py-2 px-3 text-gray-700 leading-tight
focus:outline-none focus:shadow-outline"
          onChange={strokeWidthChangeHandler}
        >
          {strokeWidthValues.map((strokeWidthValue) => (
            <option key={{ strokeWidthValue }}>{strokeWidthValue}</option>
          ))}
        </select>
      </div>
    </div>
  </div>
 );
}

export default UseStateV4;
```

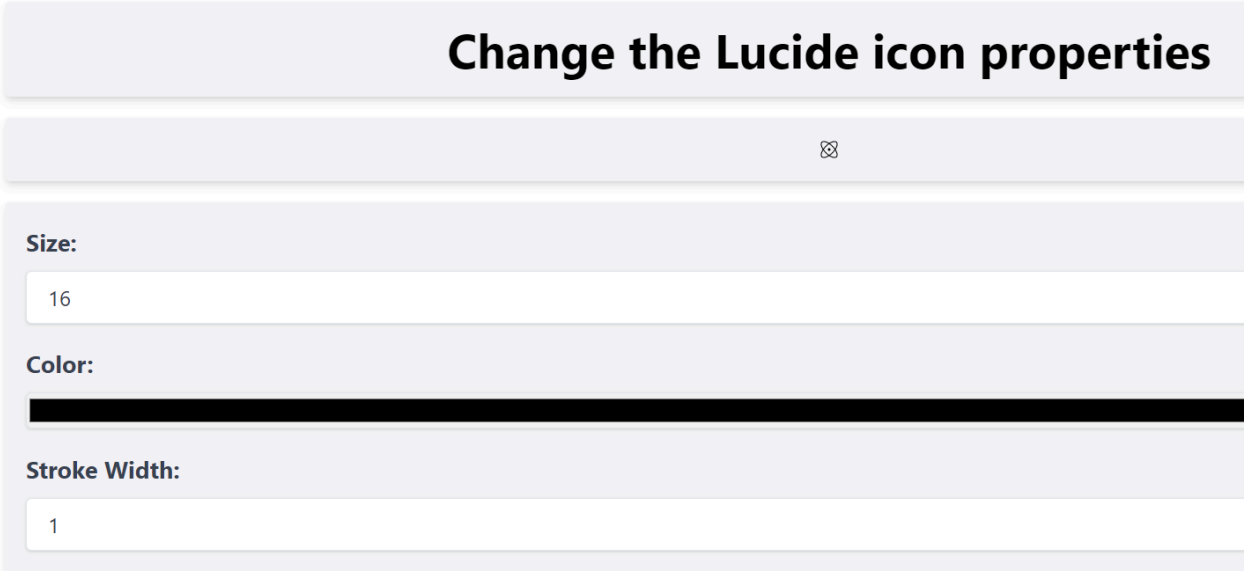
4. **Overwrite App.js** with the following code below and **save changes**.

```
import "./TW.css";
import UseStateV4 from "./components/UseStateV4";

function App() {
 return <UseStateV4></UseStateV4>;
}

export default App;
```

5. Make sure that the app is still running, if it is not running type: **npm run start** and **press enter**. Look at the browser and ensure that your output is similar below.



6. **Lucide icons** are **component** based icons, to **update the icon**, **navigate to**: https://lucide.dev/icons/ and **click** on the **icon you like** and then **click** the **Copy JSX** button. Take note of the **name** and **update import** and **component name** on the UseStateV4.js file.
7. Look at the browser and the **new icon should visible**, **change** the **value** of the **Size**, **Color** and **Stroke Width**, **take** a **screenshot** and name it: **useState4.png**

Submission:

1. Create a new folder named **react** and **copy** the **3 screenshots** and the **3 components**.
2. Create a **zip file** of the **react folder**.
3. Submit the **zip file** to **GBC - D2L**.