St. Francis Institute of Technology

(Engineering College)

An Autonomous Institute, Affiliated to University of Mumbai NAAC A+ Accredited | CMPN, EXTC, INFT NBA Accredited | ISO 9001:2015 Certified

Department of Artificial Intelligence and Machine Learning

Academic Year: 2025-2026 **Term**: Even (Jan. 2026 – Jun. 2026) **Class / Branch**: SE – AIML **Semester**: IV
Course: Web Programming Lab. (AI4VS_LR4)

Date of Assignment: / /2026 **Date of Submission:** / /2026

Pre-Lab Exercises for Experiment-8

1. Pre-Lab Activity 1: Basic useState Implementation
    a. Create a component that:
        i. Displays a message: **“Hello Student”**
        ii. Has a button labeled **“Change Message”**
        iii. On clicking the button:

    b. The message should change to **“Welcome to React State”**

CODE

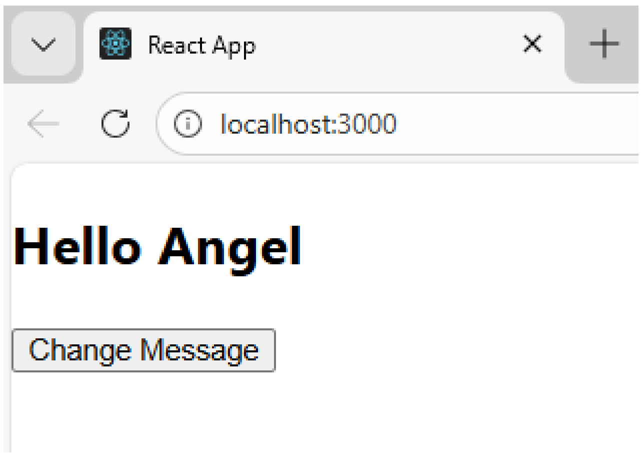
1_UseState.js

```
import React, { useState } from "react";
function UseStateExample() {
  const [message, setMessage] = useState("Hello Angel");
  const changeMessage = () => {
    setMessage("Welcome to React State");
  };
  return (
    <div>
      <h2>{message}</h2>
      <button onClick={changeMessage}>
        Change Message
      </button>
    </div>
  );
}
export default UseStateExample;
```

App.js

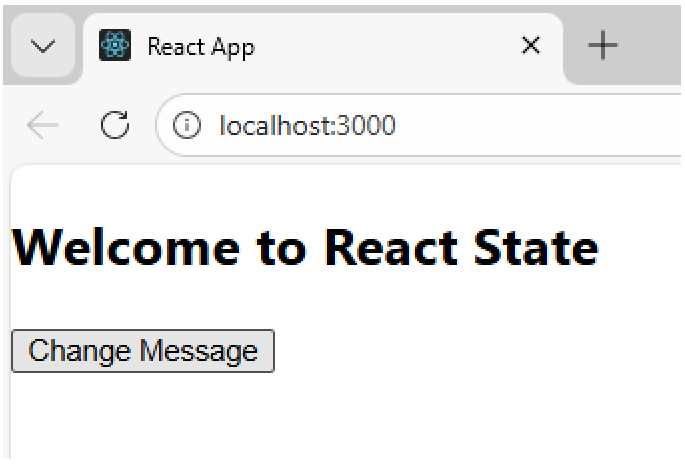
```
import React from "react";
import UseStateExample from "./1_UseState";
function App() {
  return (
    <div>
      <UseStateExample />
    </div>
  );
}
export default App;
```

OUTPUT



```
Hello Angel

Change Message
```



```
Welcome to React State

Change Message
```

2. Pre-Lab Activity 2: Simple Counter

1. Create a simple counter that:
   a. Displays a number (initial value 0)
   b. Has two buttons:
      - Increase
      - Decrease
   c. Clicking buttons updates the number accordingly

CODE

2_Counter.js

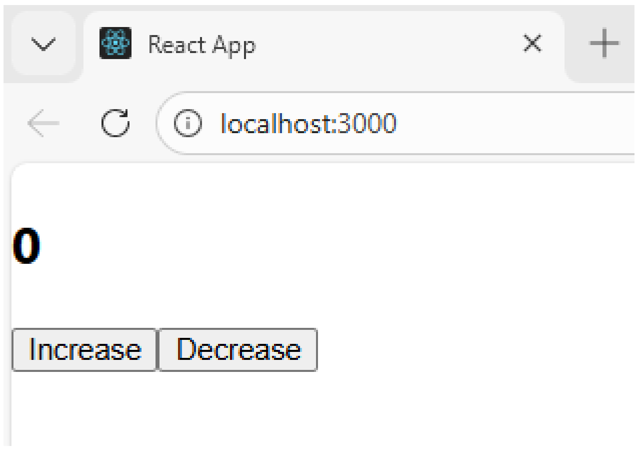
```
import React, { useState } from "react";
function Counter() {
  const [count, setCount] = useState(0);
  const increase = () => {
    setCount(count + 1);
  };
  const decrease = () => {
    setCount(count - 1);
  };
  return (
    <div>
      <h2>{count}</h2>

      <button onClick={increase}>Increase</button>
      <button onClick={decrease}>Decrease</button>
    </div>
  );
}
export default Counter;
```

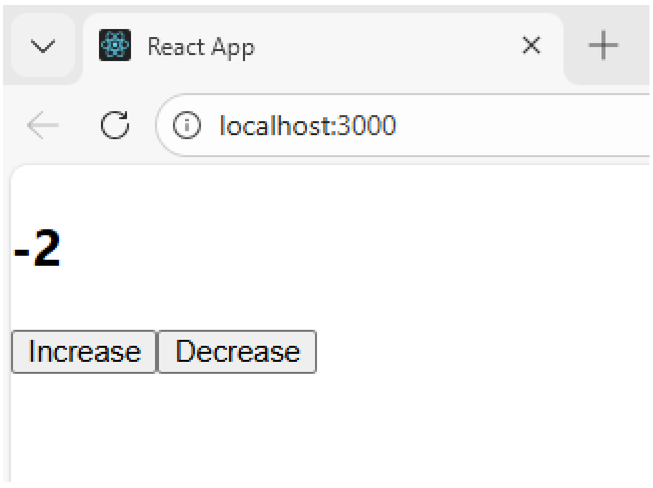
App.js

```
import React from "react";
import Counter from "./2_Counter";
function App() {
  return (
    <div>
      <Counter />
    </div>
  );
}
export default App;
```

OUTPUT



```
0
```



```
-2
```

3. Pre-Lab Activity 3: Controlled Text Input

1. Create an input field.
2. Display the entered text below the input field in real-time.
   Example:
   Input: Rahul
   Output: You entered: Rahul

CODE

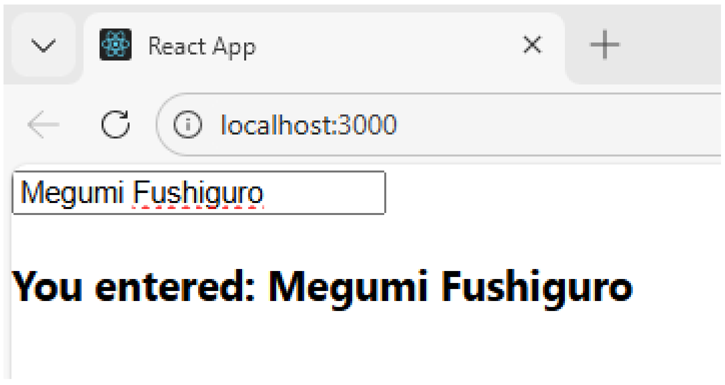
3_ControlledInput.js

```
import React, { useState } from
"react";
function ControlledInput() {
  const [text, setText] =
useState("");
  const handleChange = (event) => {
    setText(event.target.value);
  };
  return (
    <div>
      <input
        type="text"
        value={text}
        onChange={handleChange}
        placeholder="Enter your name"
      />
      <h3>You entered: {text}</h3>
    </div>
  );
}
export default ControlledInput;
```

App.js

```
import React from "react";
import ControlledInput from
"./3_ControlledInput";
function App() {
  return (
    <div>
      <ControlledInput />
    </div>
  );
}
export default App;
```

OUTPUT



```
Megumi Fushiguro

You entered: Megumi Fushiguro
```

4. Pre-Lab Activity 4: Multiple State Variables

1. Create two input fields:
   - Name
   - City
2. Display both values dynamically below.

CODE

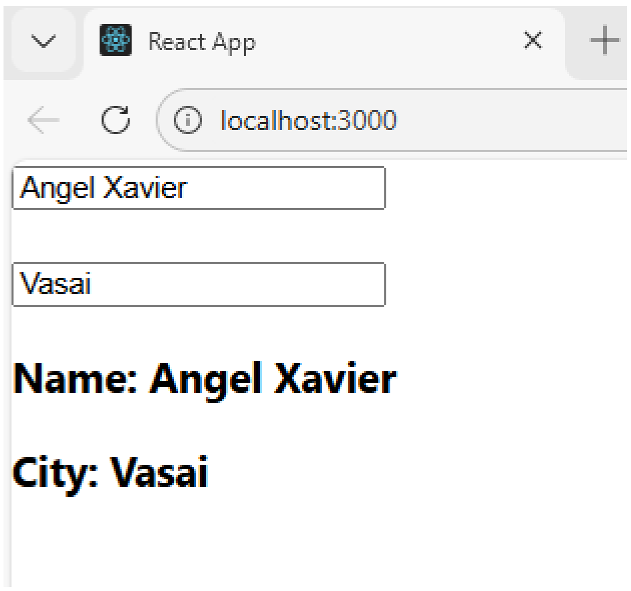
4_MultipleState.js

```
import React, { useState } from
"react";
function MultipleState() {
  const [name, setName] =
useState("");
  const [city, setCity] =
useState("");
  return (
    <div>
      <input
        type="text"
        placeholder="Enter Name"
        value={name}
        onChange={(e) =>
setName(e.target.value)}
      />
      <br /><br />
      <input
        type="text"
        placeholder="Enter City"
        value={city}
        onChange={(e) =>
setCity(e.target.value)}
      />
      <h3>Name: {name}</h3>
      <h3>City: {city}</h3>
    </div>
  );
}
export default MultipleState;
```

App.js

```
import React from "react";
import MultipleState from
"./4_MultipleState";
function App() {
  return (
    <div>
      <MultipleState />
    </div>
  );
}
export default App;
```

OUTPUT



```
Angel Xavier

Vasai

Name: Angel Xavier

City: Vasai
```

Pre-Lab Activity 5: Single State Object

Modify Activity 4 to:
1. Use only ONE state object.
2. Store both name and city inside the same object.
3. Update specific field without overwriting the other.

CODE

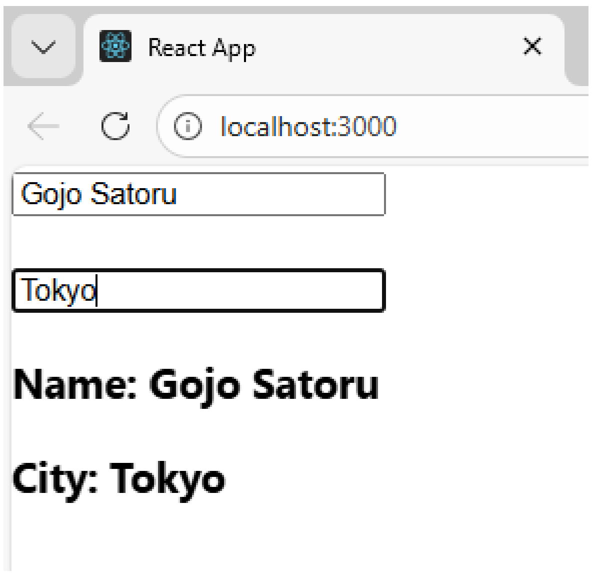
5_StateObject.js

```
import React, { useState } from
"react";
function StateObject() {
  const [formData, setFormData] =
useState({
    name: "",
    city: ""
  });
  const handleChange = (event) => {
    const { name, value } =
event.target;
    setFormData({
      ...formData,
      [name]: value
    });
  };
  return (
    <div>
      <input
        type="text"
        name="name"
        placeholder="Enter Name"
        value={formData.name}
        onChange={handleChange}
      />
      <br /><br />
      <input
        type="text"
        name="city"
        placeholder="Enter City"
        value={formData.city}
        onChange={handleChange}
      />
      <h3>Name: {formData.name}</h3>
      <h3>City: {formData.city}</h3>
    </div>
  );
}
export default StateObject;
```

App.js

```
import React from "react";
import StateObject from
"./5_StateObject";
function App() {
  return (
    <div>
      <StateObject />
    </div>
  );
}
export default App;
```

OUTPUT



```
Gojo Satoru

Tokyo

Name: Gojo Satoru

City: Tokyo
```

Pre-Lab Activity 6: Simple Validation Logic

1. Create a single input field for email.
2. When a button is clicked:
   - Show error message if field is empty.
   - Show error if email does not contain “@”.
   - Show success message if valid.

CODE

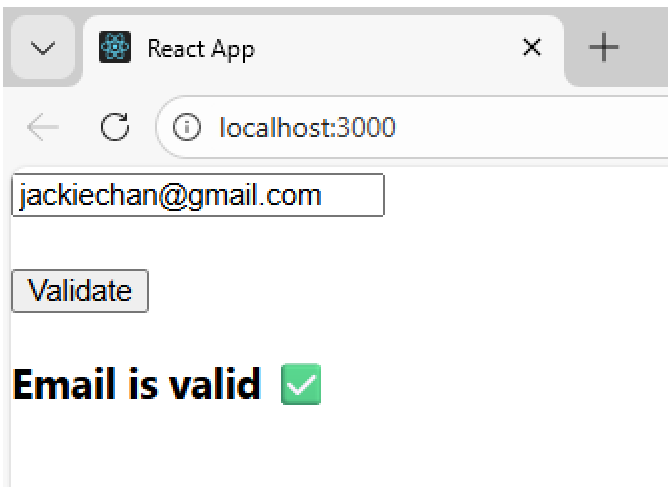
6_EmailValidation.js

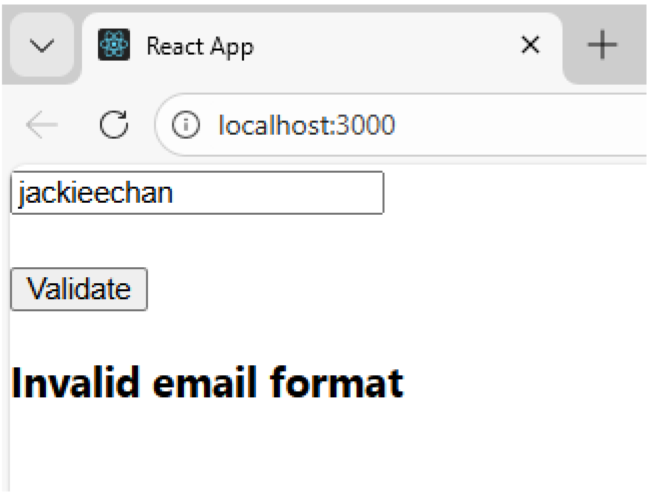
```
import React, { useState } from
"react";
function EmailValidation() {
  const [email, setEmail] =
useState("");
  const [message, setMessage] =
useState("");
  const validateEmail = () => {
    if (email === "") {
      setMessage("Email field cannot
be empty");
    } else if (!email.includes("@")) {
      setMessage("Invalid email
format");
    } else {
      setMessage("Email is valid ✅");
    }
  };
  return (
    <div>
      <input
        type="text"
        placeholder="Enter email"
        value={email}
        onChange={(e) =>
setEmail(e.target.value)}
      />
      <br /><br />
      <button onClick={validateEmail}>
        Validate
      </button>
      <h3>{message}</h3>
    </div>
  );
}
export default EmailValidation;
```

App.js

```
import React from "react";
import EmailValidation from
"./6_EmailValidation";
function App() {
  return (
    <div>
      <EmailValidation />
    </div>
  );
}
export default App;
```

OUTPUT


React App
localhost:3000
jackiechan@gmail.com
Validate
Email is valid ✅


React App
localhost:3000
jackieechan
Validate
Invalid email format

Pre-Lab Questions

1. What is state in React?
Ans. State is a built-in React object used to store dynamic data in a component.When state changes, React re-renders the component automatically.

2. What is the difference between normal variable and state variable?
Ans. **Normal variable:** Does not trigger re-render when changed.
State variable: Managed by React using `useState`. When updated, it causes the component to re-render.

3. What is a controlled component?
Ans. A controlled component is a form element (like input) whose value is controlled by React state using `useState` and `onChange`.

4. Why does React re-render when state changes?
Ans. React re-renders because state represents dynamic UI data.When state updates, React updates the Virtual DOM and refreshes the UI accordingly.

5. Why do we use the spread operator in object state?

Ans. The spread operator (`...`) is used to copy existing state values while updating a specific property. It prevents overwriting other properties in the object.