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-9

Pre-Lab Activity 1: Understanding useEffect Execution

Task:
1. Create a functional component.
2. Display a simple message on screen.
3. Use `useEffect` to:
   - Print a message in the console when the component loads.

CODE

1.js

```
import React, { useEffect } from "react";
function Welcome() {
  useEffect(() => {
    console.log("Component loaded successfully!");
  }, []);
  return (
    <div>
      <h2>Pre-Lab Activity 1</h2>
      <p>This message is displayed using a functional component.</p>
    </div>
  );
}
export default Welcome;
```

App.js

```
import Welcome from "./1";
function App() {
  return <Welcome />;
}
export default App;
```

OUTPUT

Pre-Lab Activity 2: Dependency Array Behavior

Task:

1. Create a counter using useState.
2. Add a useEffect that:
   - Logs “Counter Updated” whenever count changes.
3. Experiment with:
   - No dependency array
   - Empty dependency array
   - Dependency with count

CODE

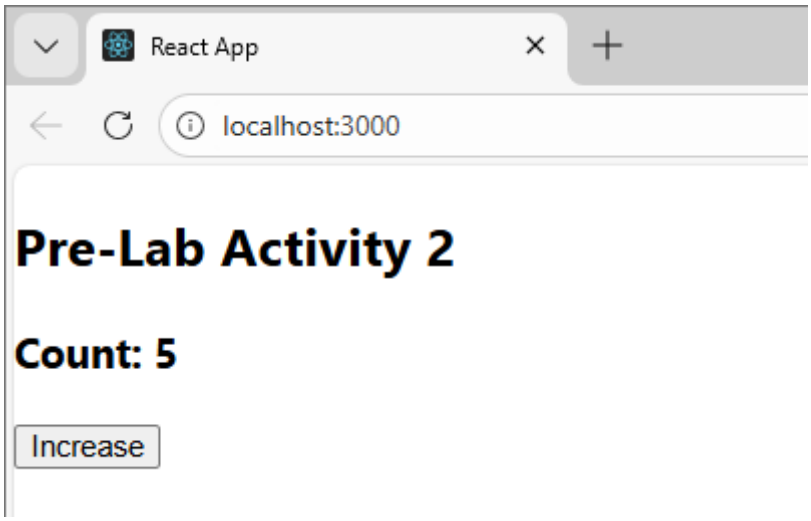
2.js

```
import React, { useState, useEffect }
from "react";
function Welcome() {
  const [count, setCount] =
useState(0);
  // 🔴 No dependency array → runs on
every render
  useEffect(() => {
    console.log("No dependency array →
runs every render");
  });
  // 🟢 Empty dependency → runs once
on load
  useEffect(() => {
    console.log("Empty dependency →
runs once");
  }, []);
  // 🔵 Dependency with count → runs
when count changes
  useEffect(() => {
    console.log("Counter Updated");
  }, [count]);
  return (
    <div>
      <h2>Pre-Lab Activity 2</h2>
      <h3>Count: {count}</h3>
      <button onClick={() =>
setCount(count + 1)}>
        Increase
      </button>
    </div>
  );
}
export default Welcome;
```

App.js

```
import Welcome from "./2";
function App() {
  return <Welcome />;
}
export default App;
```

OUTPUT



Pre-Lab Activity 3: Simulating API Call using setTimeout

Before calling real APIs, simulate asynchronous behavior.

Task:

1. Create a component.
2. Create a state called `data`.
3. Use useEffect to:
   - Simulate fetching data using setTimeout (2 seconds delay).
   - After delay, update state with sample data.
4. Show "Loading…" until data appears.

CODE

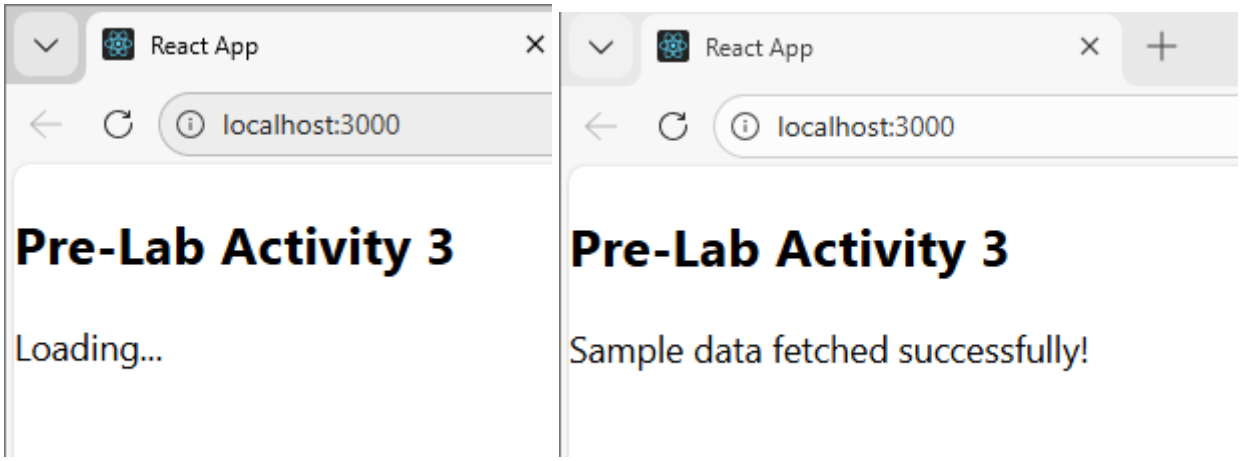
3.js

```
import React, { useState, useEffect }
from "react";
function Welcome() {
  const [data, setData] =
useState(null);
  useEffect(() => {
    // simulate API call (2 sec delay)
    const timer = setTimeout(() => {
      setData("Sample data fetched
successfully!");
    }, 2000);
    // cleanup (good practice)
    return () => clearTimeout(timer);
  }, []);
  return (
    <div>
      <h2>Pre-Lab Activity 3</h2>
      {data ? (
        <p>{data}</p>
      ) : (
        <p>Loading...</p>
      )}
    </div>
  );
}
export default Welcome;
```

App.js

```
import Welcome from "./3";
function App() {
  return <Welcome />;
}
export default App
```

OUTPUT



Pre-Lab Activity 4: Basic Fetch API (Without useEffect)

Task:

1. Create a button labeled “Fetch Data”.

2. On button click:
   - Fetch data from a public API.
   - Print response in console.

CODE

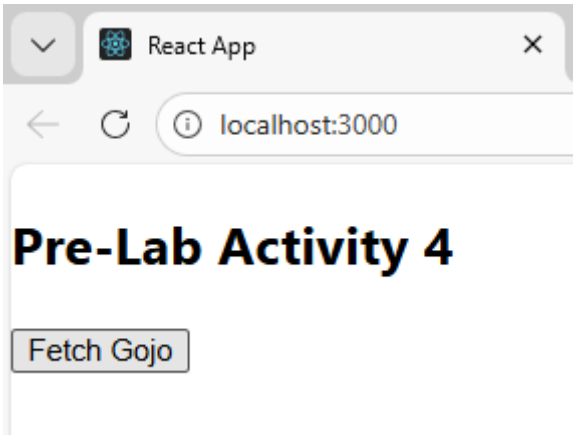
4.js

```
import React from "react";
function Welcome() {
  const fetchData = async () => {
    try {
      const response = await
fetch("https://jsonplaceholder.typicod
e.com/posts/1");
      const data = await
response.json();
      console.log("Fetched Gojo:",
data);
    } catch (error) {
      console.log("Error:", error);
    }
  };
  return (
    <div>
      <h2>Pre-Lab Activity 4</h2>
      <button onClick={fetchData}>
        Fetch Gojo
      </button>
    </div>
  );
}
export default Welcome;
```

App.js

```
import Welcome from "./4";
function App() {
  return <Welcome />;
}
export default App;
```

OUTPUT




```
Download the React DevTools for a better development experience:
https://react.dev/link/react-devtools
Fetched Gojo:                                                    4.js:8
▶ {userId: 1, id: 1, title: 'sunt aut facere repellat provident occaecati e
  xcepturi optio reprehenderit', body: 'quia et suscipit\nsuscipit recusand
  ae consequuntur …strum rerum est autem sunt rem eveniet architecto'}
```

Pre-Lab Activity 5: Async/Await Practice

Task:

1. Create an async function.
2. Use async/await instead of .then().
3. Log fetched data in console.
4. Add try-catch for error handling.

CODE

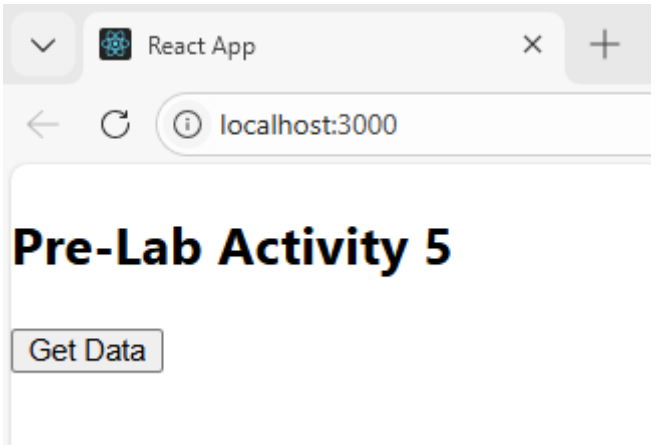
5.js

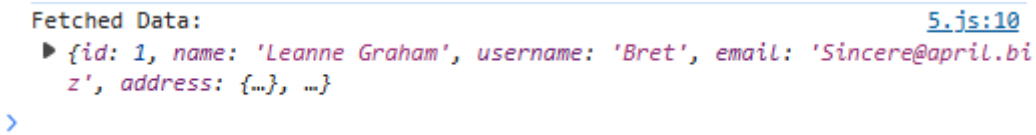
```
import React from "react";
function Welcome() {
  const getData = async () => {
    try {
      const response = await
fetch("https://jsonplaceholder.typicod
e.com/users/1");
      const data = await
response.json();
      console.log("Fetched Data:",
data);
    } catch (error) {
      console.log("Error:", error);
    }
  };
  return (
    <div>
      <h2>Pre-Lab Activity 5</h2>
      <button onClick={getData}>
        Get Data
      </button>
    </div>
  );
}
export default Welcome;
```

App.js

```
import Welcome from "./5";
function App() {
  return <Welcome />;
}
export default App;
```

OUTPUT





Pre-Lab Activity 6: Rendering Array Data Dynamically

Task:

1. Create an array of objects manually.
2. Store it in state.
3. Use map() to display:
   - Name
   - Email
4. Render items in list format.

CODE

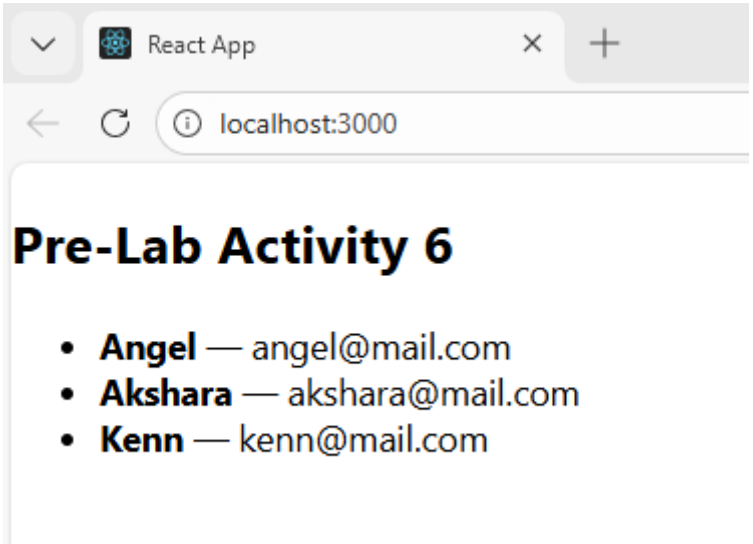
6.js

```
import React, { useState } from "react";
function Welcome() {
  const [users] = useState([
    { id: 1, name: "Angel", email: "angel@mail.com" },
    { id: 2, name: "Akshara", email: "akshara@mail.com" },
    { id: 3, name: "Kenn", email: "kenn@mail.com" }
  ]);
  return (
    <div>
      <h2>Pre-Lab Activity 6</h2>
      <ul>
        {users.map(user => (
          <li key={user.id}>
<strong>{user.name}</strong> — {user.email}
          </li>
        ))}
      </ul>
    </div>
  );
}
export default Welcome;
```

App.js

```
import Welcome from "./6";
function App() {
  return <Welcome />;
}
export default App;
```

OUTPUT



Pre-Lab Record Questions

1. When does useEffect execute?
Ans. useEffect runs after a component renders. Without dependencies it runs on every render, with an empty array it runs once, and with dependencies it runs when those values change.

2. What is the difference between .then() and async/await?
Ans. .then() handles promises using chaining, which can become complex. async/await provides cleaner, synchronous-like code and improves readability while doing the same task.

3. Why is loading state important during API calls?
Ans. A loading state informs users that data is being fetched. It improves user experience by preventing blank screens and showing progress.

4. Why must fetched data be stored in state?
Ans. Fetched data is stored in state so React can detect changes and re-render the UI. This ensures the latest data is displayed on the screen.