WEEK-5

AIM:A) Write a program to render HTML to a web page.

DESCRIPTION:

In React, HTML is rendered using JSX, which looks like HTML but is written inside JavaScript.
JSX allows embedding HTML-like tags such as <h1>, <p>, and directly in code.
We display this content in the browser using createRoot().render() (React 18) or ReactDOM.render() (React 17).
This makes creating dynamic web pages simple and efficient.

SOURCE CODE 1:-

(Index.html)

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Render HTML Demo</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <header>
    <h2>24A95A6103</h2>
    <h1>Render HTML to a Web Page</h1>
  </header>

  <main>
    <section class="card">
      <h2 id="greeting">Hello, visitor!</h2>
      <p id="description">Click the button to update this message.</p>

      <div class="controls">
        <input id="nameInput" type="text" placeholder="Type your name" />
        <button id="sayHelloBtn">Say Hello</button>
        <button id="resetBtn">Reset</button>
      </div>

      <div class="info">
        <strong>Array of items rendered below:</strong>
        <ul id="itemList"></ul>
      </div>
    </section>
  </main>

  <script src="script.js"></script>
</body>
</html>
```

SOURCE CODE 2:-
(Script.js)

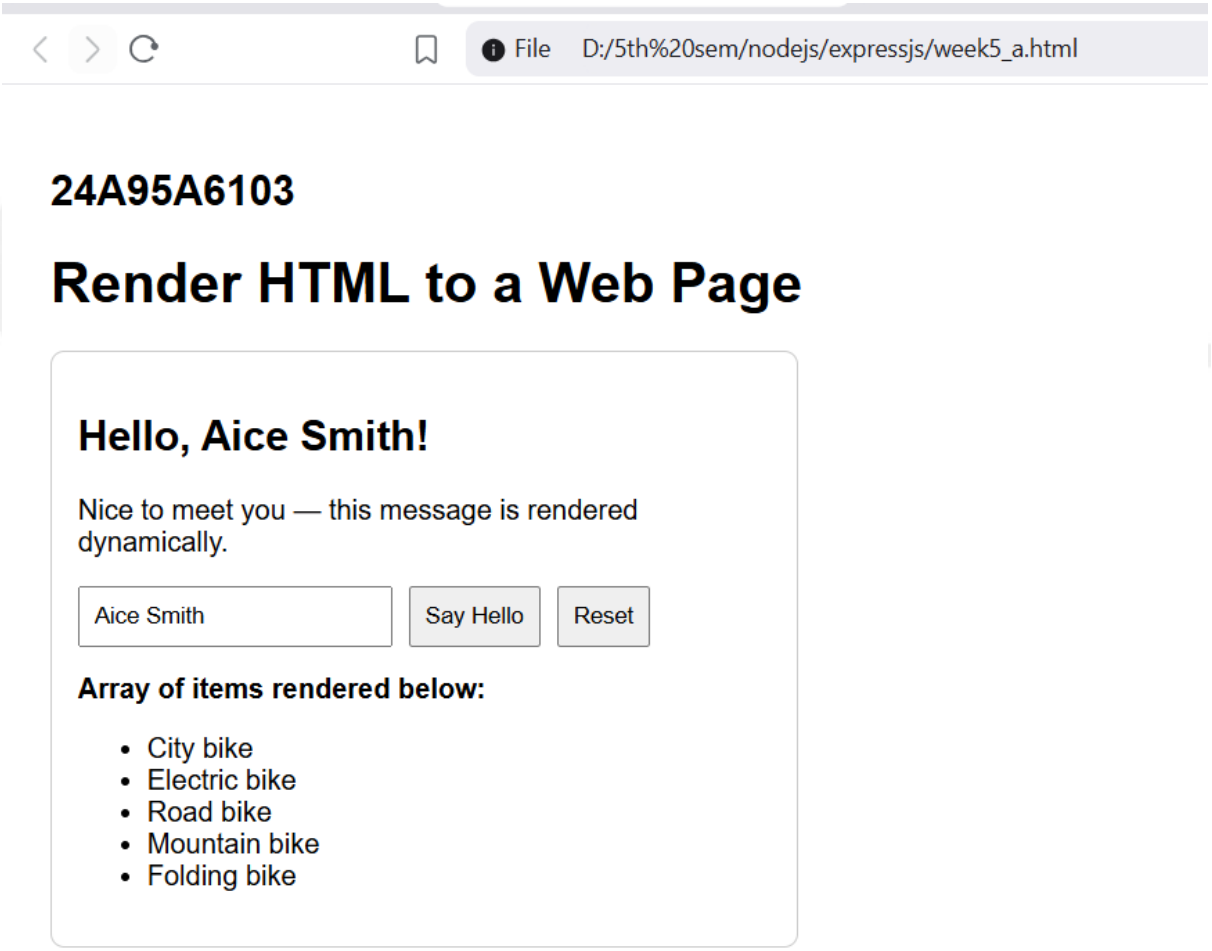
```
// Sample data to render as a list
const items = ["City bike", "Electric bike", "Road bike", "Mountain bike", "Folding bike"];

// Render list into #itemList
const itemList = document.getElementById("itemList");
items.forEach(item => {
  const li = document.createElement("li");
  li.textContent = item;
  itemList.appendChild(li);
});

// UI elements
const nameInput = document.getElementById("nameInput");
const sayHelloBtn = document.getElementById("sayHelloBtn");
const resetBtn = document.getElementById("resetBtn");
const greeting = document.getElementById("greeting");
const description = document.getElementById("description");

// Event handlers
sayHelloBtn.addEventListener("click", () => {
  const name = nameInput.value.trim();
  if (name) {
    greeting.textContent = `Hello, ${name}!`;
    description.textContent = "Nice to meet you — this message is rendered dynamically.";
  } else {
    greeting.textContent = "Hello, visitor!";
    description.textContent = "Please type your name and click 'Say Hello'.";
  }
});

resetBtn.addEventListener("click", () => {
  nameInput.value = "";
  greeting.textContent = "Hello, visitor!";
  description.textContent = "Click the button to update this message.";
});
```

SOURCE CODE 3:-
(Styles.css)

```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

.card {
  border: 1px solid #ccc;
  padding: 15px;
  border-radius: 8px;
  max-width: 400px;
```

```
}

.controls {
  margin-top: 10px;
}

.controls input,
.controls button {
  padding: 8px;
  margin-right: 5px;
}

.info {
  margin-top: 15px;
}
```

OUTPUT:



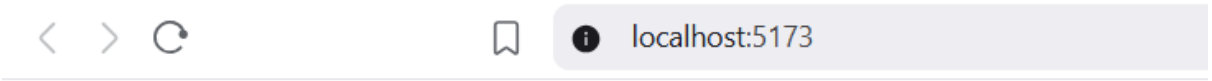
AIM:B) Write a program for writing markup with JSX.

DESCRIPTION:
JSX in React allows you to write HTML-like markup directly inside JavaScript. You can embed JavaScript expressions using {} to display dynamic content, making your UI interactive. React then converts this JSX into regular JavaScript that the browser can render, which makes building and maintaining user interfaces simple and readable.

SOURCE CODE:
(App.jsx)

```
import React from "react";
function App(){
  const name = "Alice Smith";
  const year = new Date().getFullYear();

  return(
    <div>
      <h1>24A95A6103</h1>
      <h1>Welcome, {name}</h1>
      <p>Current Year: {year}</p>
      <p>2+3 = {2+3}</p>
    </div>
  );
}

export default App;
```


OUTPUT:



24A95A6103

Welcome, Alice Smith

Current Year: 2025

2+3 = 5

AIM:C) Write a program for creating and nesting components (function and class).

DESCRIPTION:

In React, components are the building blocks of UI.

- Function Components → Defined using functions, simpler and preferred.
- Class Components → Defined using ES6 classes, used before React Hooks but still valid.

Nesting Components means putting one component inside another to build a hierarchy.
Example:
App component (main parent) Contains Header (function component)
Contains Content (class component)
Inside Content, we nest another Footer (function component).

SOURCE CODE 1:-
(App.jsx)

```
import React from "react";
import Header from "./components/Header";
import Footer from "./components/Footer";

function App() {
  return (
    <div style={{ padding: "20px", fontFamily: "Arial, sans-serif" }}>
      <h1>React Components Example</h1>
      <h4>Name: Naga Veeranna</h4>
      <h5>Roll Number: 24A95A6103</h5>

      {/* Nested Components */}
      <Header />
      <Footer />
    </div>
  );
}

export default App;
```

SOURCE CODE 2:-
(Header.jsx)

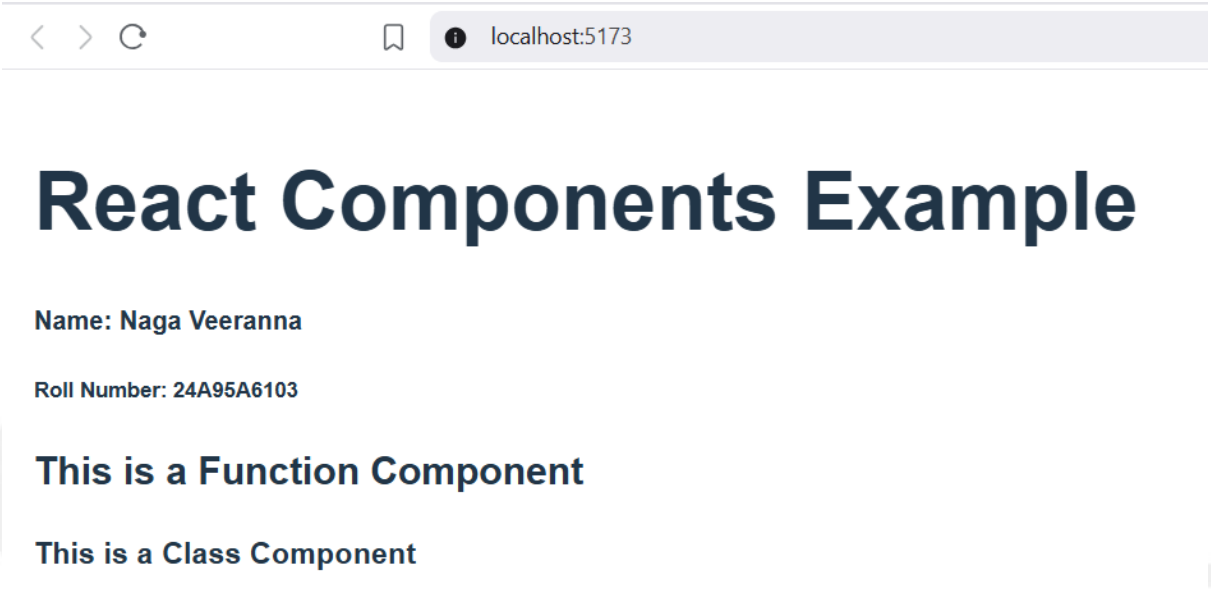
```
import React from "react";

// Function Component
function Header() {
  return <h2>This is a Function Component</h2>;
}

export default Header;
```

SOURCE CODE 3:-
(Footer.jsx)

```
import React, { Component } from "react";
```

```
// Class Component
class Footer extends Component {
  render() {
    return <h3>This is a Class Component</h3>;
  }
}

export default Footer;
```

OUTPUT:



WEEK-6

AIM:A) Write a program to work with props and states.

DESCRIPTION:

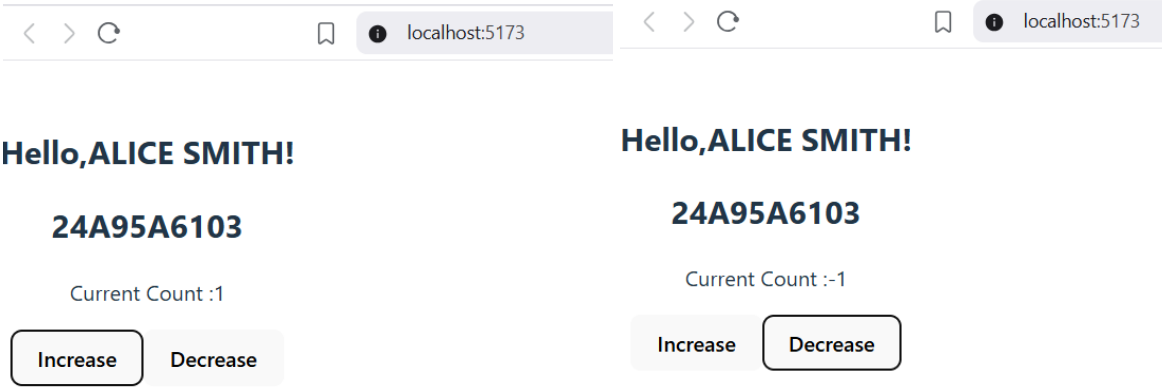
In React, props are used to pass data from a parent component to a child component, making components reusable and dynamic. State is used to store data that can change over time within a component, allowing the UI to update automatically when the state changes. Using the useState hook, functional components can manage their state, while props remain read-only and state can be updated with setter functions to re-render the component. This combination of props and state makes React applications interactive and dynamic.

SOURCE CODE:

(app.jsx)

```
import React, {useState} from "react";
function Greeting(props){
  return <h2> Hello,{props.name}!</h2>;
}
export default function App(){
  const [count,setCount] = useState(0);
  return (
    <div style={{textAlign: "center",marginTop: "50px"}}>
      <Greeting name = "ALICE SMITH" />
      <h2>24A95A6103</h2>
      <p> Current Count :{count}</p>
      <button onClick={() => setCount(count + 1)}> Increase</button>
      <button onClick={() => setCount(count - 1)}> Decrease</button>
    </div>
  );
}
```

OUTPUT:



AIM:B) Write a program to add styles (CSS & Sass Styling) and display data.

DESCRIPTION:
In React, you can add styles using CSS or Sass (SCSS) to make your components visually appealing. CSS files allow you to define classes and apply them to elements, while Sass adds features like variables, nesting, and functions for more organized and reusable styling. By importing these style files into your component, you can style elements such as headings, paragraphs, and containers while displaying dynamic data, keeping your design separate from your component logic.

SOURCE CODE 1:-
(App.jsx)

```
import React from "react";
import "./App.css";
export default function App(){
  const users = [
    {id: 1, name: "JOHN", age:18},
    {id: 2, name: "ALICE", age:19},
    {id: 3, name: "SMITH", age:20},
  ];
  return (
    <div className="container">
      <h1 className="title"> User list</h1>
      <h3>24A95A6103</h3>
      {users.map((user) => (
        <div className="card" key={user.id}>
          <h3> {user.name}</h3>
          <p>Age: {user.age}</p>
        </div>
      ))}
    </div>
  );
}
```

SOURCE CODE 2:-

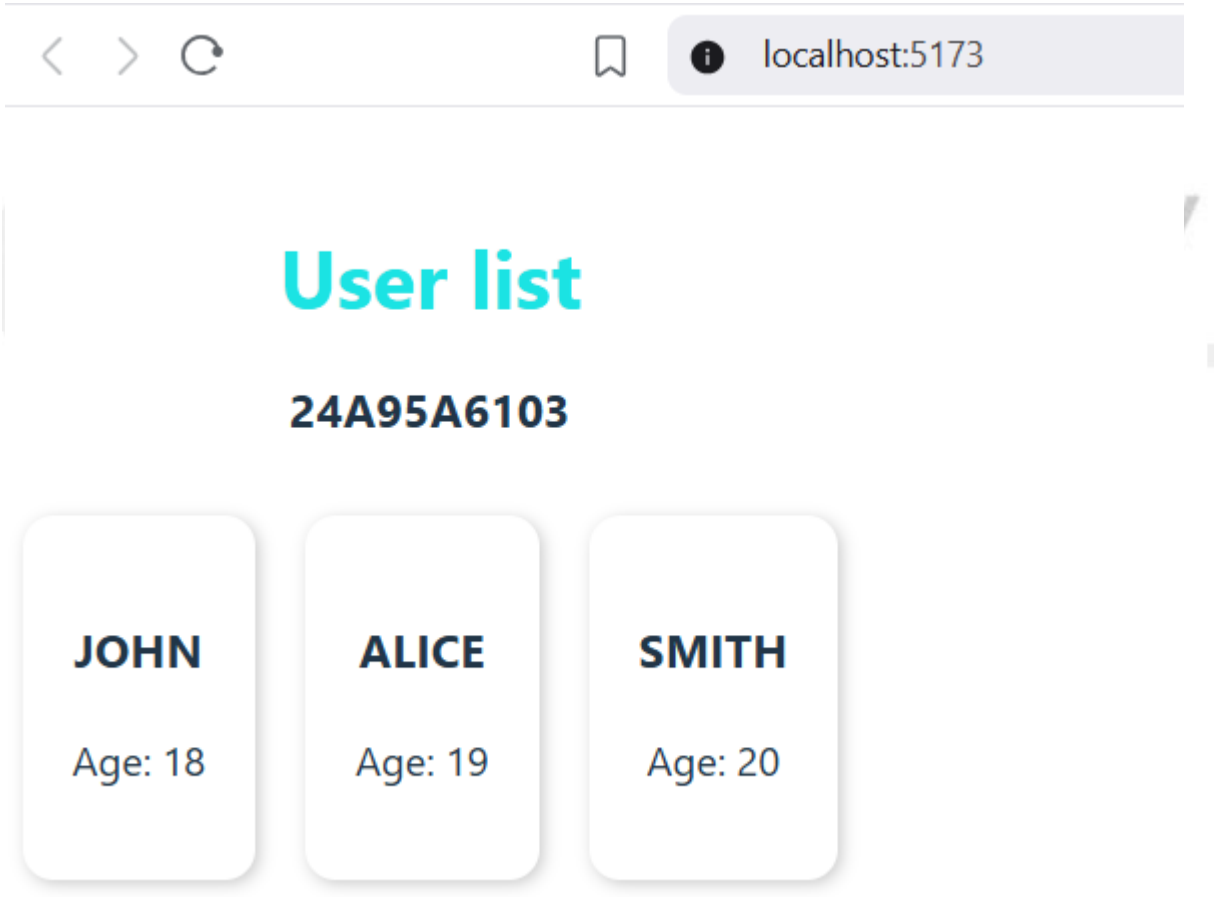
```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

SOURCE CODE 3:-

```
.container{
```

```
  text-align : center;
  margin-top: 50px;
}
.title{
  color: #1be3e3ff;
  font-size: 2rem;
  font-weight:bold;
}
.card{
  display:inline-block;
  background: #ffffff;
  padding: 20px;
  margin: 10px;
  border-radius: 12px;
  box-shadow: 2px 2px 6px rgba(0, 0 , 0, 0.2)
}
```

OUTPUT:



AIM:C) Write a program for responding to events.

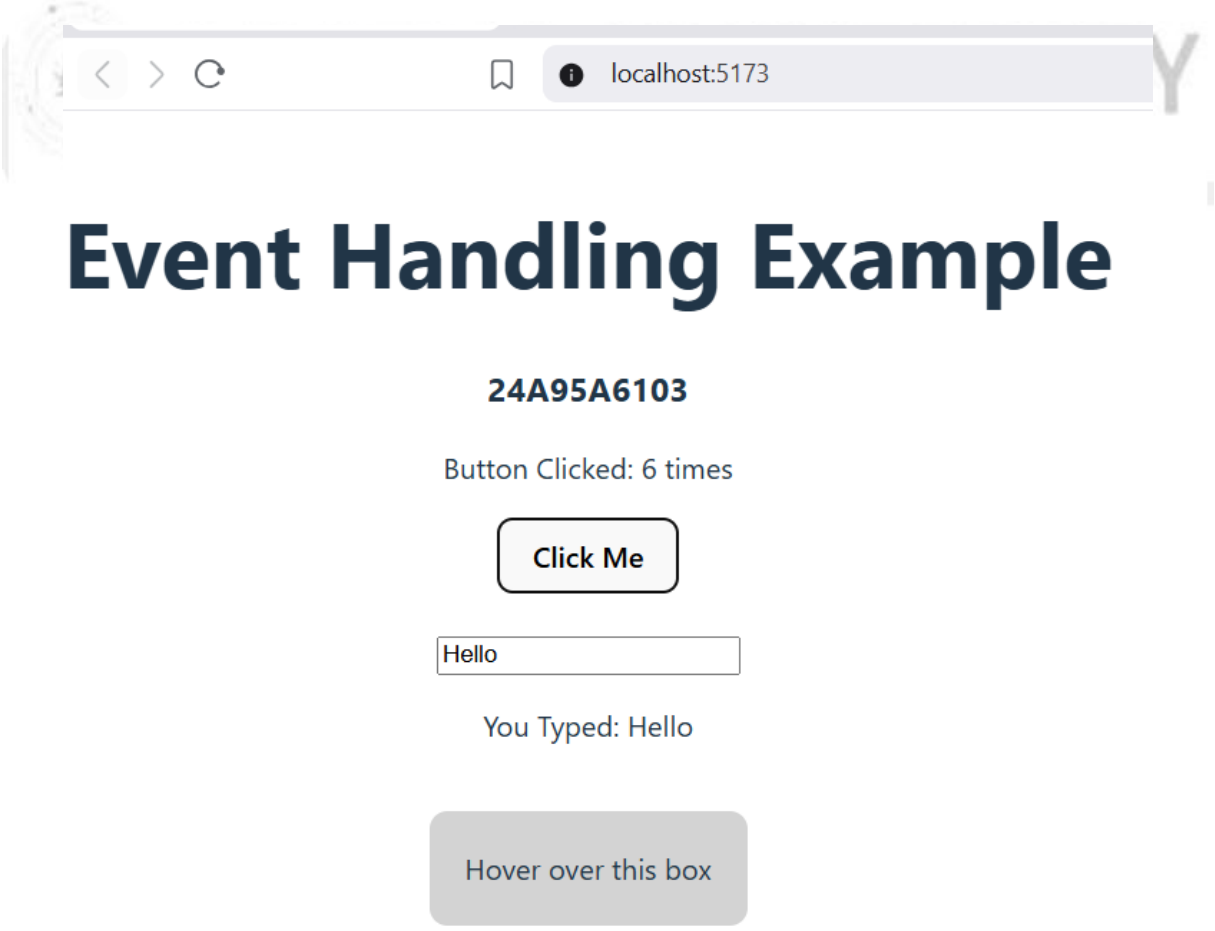
DESCRIPTION:
In React, components can respond to user events such as clicks, mouse movements, or key presses by defining event handler functions. Events are attached to elements using attributes like onClick, onChange, or onSubmit, and when the event occurs, the corresponding function executes, often updating the component's state. This allows the UI to dynamically react to user interactions, making web applications

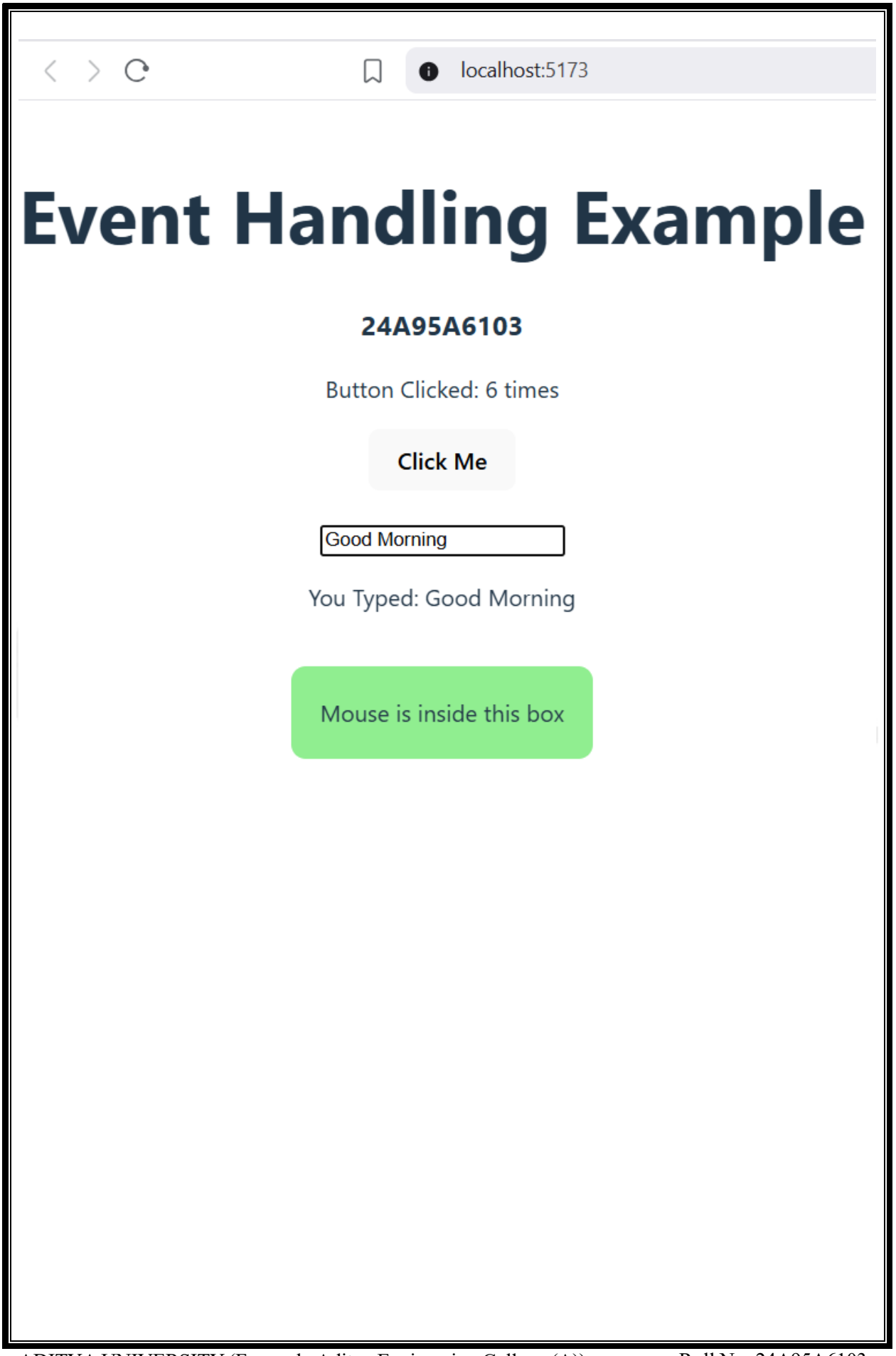
SOURCE CODE:

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

export default function App() {
  const [count, setCount] = useState(0);
  const [message, setMessage] = useState("");
  const [hovered, setHovered] = useState(false);

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

  const handleInputChange = (event) => {
    setMessage(event.target.value);
  };

  const handleMouseEnter = () => {
    setHovered(true);
  };

  const handleMouseLeave = () => {
    setHovered(false);
  };

  return (
    <div style={{ textAlign: "center", marginTop: "50px" }}>
      {/* Click Event */}
      <h1>Event Handling Example</h1>
      <h3>24A95A6103</h3>
      <p>Button Clicked: {count} times</p>
      <button onClick={handleClick}>Click Me</button>

      {/* Input Event */}
      <div style={{ marginTop: "20px" }}>
        <input
          type="text"
          placeholder="Type Something..."
          value={message}
          onChange={handleInputChange}
        />
        <p>You Typed: {message}</p>
```

```
    </div>

    {/* Mouse Events */}
    <div
      style={{
        marginTop: "20px",
        padding: "20px",
        backgroundColor: hovered ? "lightgreen" : "lightgray",
        borderRadius: "10px",
        display: "inline-block",
        cursor: "pointer",
      }}
      onMouseEnter={handleMouseEnter}
      onMouseLeave={handleMouseLeave}
    >
      {hovered ? "Mouse is inside this box" : "Hover over this box"}
    </div>
  </div>
 );
}
```

OUTPUT:




localhost:5173
Event Handling Example
24A95A6103
Button Clicked: 6 times
Click Me
Good Morning
You Typed: Good Morning
Mouse is inside this box