Mini Project Report - 08

Master of Computer Application – General

Semester – I

Sub: FRONT_END FRAMEWORKS & TECHNOLOGIES

Topic: Number Counting System

By
Name: SUHAS D J
Reg no.: PROV/ASAC/MCA/25/7/050

Faculty Name: VEERA RAGHAV K

Faculty Signature: _____

Department of Computer Application
Alliance University
Chandapura - Anekal Main Road, Anekal
Bengaluru - 562 106

October 2025

LIST OF TABLES

INTRODUCTION

The Counting Buffer project is a beginner-friendly React application developed to understand and implement the concept of state management using the useState Hook. In modern web development, React plays a vital role in building dynamic and interactive user interfaces, and this project serves as an excellent example of how simple logic can be efficiently handled through React's features.

In this project, the application allows users to increase, decrease, or reset a numerical value displayed on the screen. The useState Hook is used to store and update the counter value whenever a button is clicked. Each button—Increase, Decrease, and Clear—is linked to a specific event handler function that modifies the state accordingly. The project also includes a well-structured CSS file that enhances the visual appeal with a neat layout, color schemes, and button styles.

By developing this project, we gain practical experience in core React concepts such as component creation, state management, event handling, JSX syntax, and integrating external CSS for styling. It also helps us understand how small modular components can be combined to form a fully functional web interface.

INPUT CODE

REACT CODE

```
import React,{useState} from'react';//importing use state
import"../src/counting.css"//for linking css file
//parent react function
function App(){
  const[count,setCount]=useState(0);//this is the syntax of usestate
  //Arrow function
  const increment=()=>{
    setCount(count+1)
  };
  const Decrement=()=>{
    setCount(count -1)
  };
  const clear=()=>{
    setCount(0)
  };
  return(
    <div>
    <h1 className="heading">Counting Buffer</h1>
    <p className="count">Counting:{count}</p>
    <button onClick={increment}className="b1">Increase</button>
        <button onClick={Decrement}className="b2">Decrement</button>
         <button onClick={clear}className="b3">Clear</button>

    </div>
  );
```

```
};
export default App;
```

CSS CODE

```
.heading{
    text-align: center;
    background-color: yellow;
    font-size: 50px;
    font-family: Arial, sans-serif;
}
p{
    text-align: center;
    font-size: 30px;
}
.b1{
    background-color: red;
    border-radius: 5px;
    border: none;
    margin-left:750px;
    font-size: 30px;
}
.b2{
    background-color: gray;
    border-radius: 5px;
    border: none;
    font-size: 30px;
    margin-left: 5px;
}
.b3{
    background-color: skyblue;
    border-radius: 5px;
    border: none;
    font-size: 30px;
    margin-left: 5px;
}
```

OUTPUT

CONCLUSION

In conclusion, the **Counting Buffer project** successfully demonstrates the power and simplicity of React in handling real-time updates to the user interface. Through this mini-project, we learned how to use the useState Hook effectively to manage and modify component states. We also explored how user actions, like clicking buttons, can trigger specific functions to update data dynamically without reloading the page.

The project provided valuable insights into the **reactive nature of React**, where changes in state automatically re-render components to reflect the latest data. Additionally, the inclusion of CSS enhanced the user experience by providing a clean and attractive interface. Overall, this project strengthened the understanding of fundamental React concepts, preparing us for more advanced projects involving complex logic, multiple components, and API integration.

.