Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of Computer Engineering and Technology
Academic Year: 2023-2024 Sem V

Fullstack Development

Lab Assignment : 05

Title: Design and develop an interactive user interface using React.

Prepared By
Saurabh Jitendra Jadhav
Roll No:-PA12
Batch A1
November 20,2023

Aim: Design and develop an interactive user interface using React.

Objectives:

1. Articulate what React is and why it is useful.
2. Explore the basic architecture of a React application.
3. Use React components to build interactive interfaces

Theory:

1. What is React? Steps to run React app using create-react-app.

->React is a JavaScript library for building user interfaces. It is declarative, efficient, and flexible. React makes it easy to create interactive UIs by using a component-based approach.

To run a React app using create-react-app, follow these steps:

1. Install Node.js and npm if you don't already have them.
2. Open a terminal window and navigate to the directory where you want to create your React app.
3. Run the following command:

npx create-react-app my-app(App Name)

This will create a new directory called **my-app** with all the files you need to start a React project.

1) Change into the my-app directory:

cd my-app

2) Start the development server:

npm start

This will start a server on your local machine at **http://localhost:3000**. You can now open this URL in your web browser to see your React app.

3)To build your React app for production, run the following command:

npm run build

2. Passing data through props.
Ans:-
Theory:
In React, components are the building blocks of a user interface. They are modular, reusable pieces of code that encapsulate a part of the UI. To enable communication between these components, React uses a mechanism called "props" (short for properties).

Props:

- **Definition:** Props are a system for passing data from a parent component to a child component.
- **Usage:** Props are passed as attributes when a component is used in JSX.
- **Nature:** Props are immutable in the child component. They are read-only and cannot be modified directly by the child.

Ex. parent component (ParentComponent) passing a message to a child component (ChildComponent) using props.

ParentComponent.js:

```
import React from 'react';
import ChildComponent from './ChildComponent';

function ParentComponent() {
  const messageToChild = "Hello from Parent!";

  return (
    <div>
      <ChildComponent message={messageToChild} />
    </div>
  );
}

export default ParentComponent;
```

ChildComponent.js:

```
import React from 'react';

function ChildComponent(props) {
  return (
    <div>
      <p>{props.message}</p>
    </div>
  );
}

export default ChildComponent;
```

Explanation:

- ParentComponent defines a variable messageToChild with the value "Hello from Parent!"
- It renders ChildComponent and passes the message as a prop: <ChildComponent message={messageToChild} />.
- ChildComponent receives the prop as an argument (props) and displays the message in a paragraph elements.

When ParentComponent is rendered, it will display the content of ChildComponent with the message received as a prop.

The output will be:

Hello from Parent!

FAQ:

1. What are React states and hooks?
-> **React States:**
In React, state is a built-in object that represents the mutable data of a component. It is used to manage and store component-specific data that can change over time. State allows React components to keep track of dynamic information, such as user input, network responses, or any data that may change during the lifetime of the component.

Key Characteristics of React State:

- Mutable: Unlike props, state can be modified by the component itself. It's intended for data that will change during the component's lifecycle.

- Local to Component: Each component can have its own state, making it local and isolated from the state of other components.

- Async Updates: State updates may be asynchronous to ensure optimal performance. React may batch multiple setState calls into a single update for efficiency.

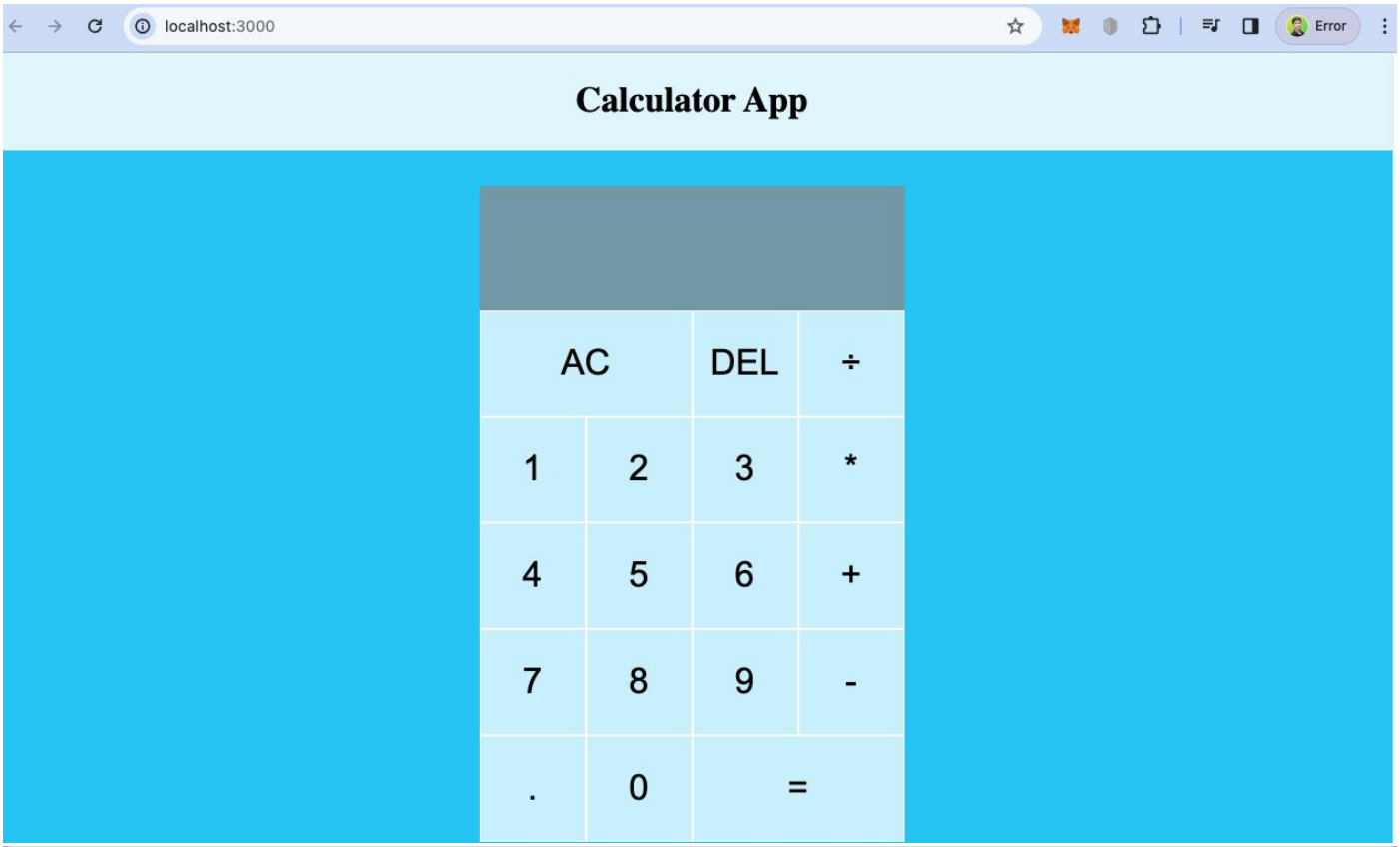
React Hooks:

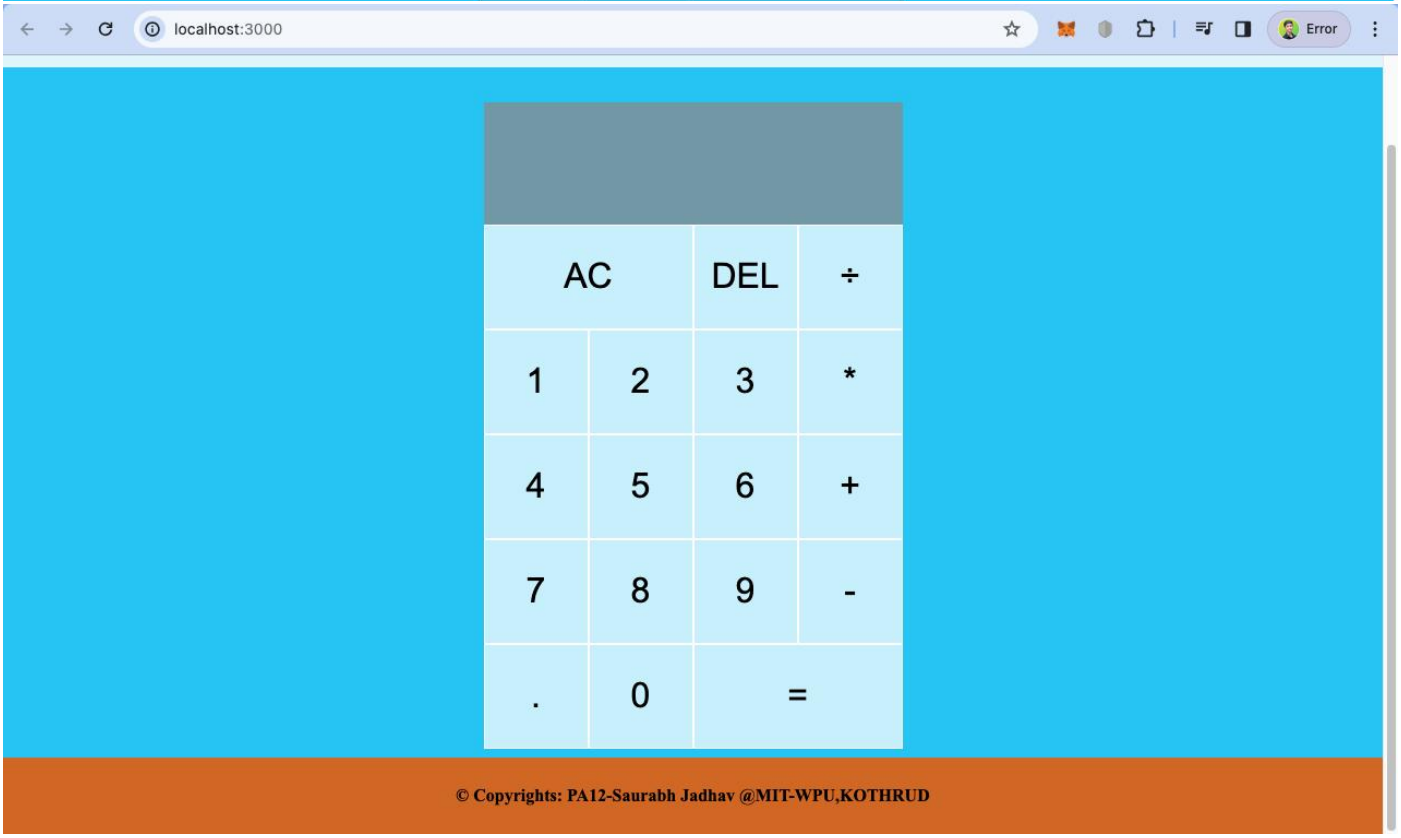
React Hooks are functions that allow functional components to use state and other React features that were previously only available in class components. Hooks were introduced in React 16.8 to enable the use of state and lifecycle features in functional components without the need for class components.

Commonly Used Hooks:

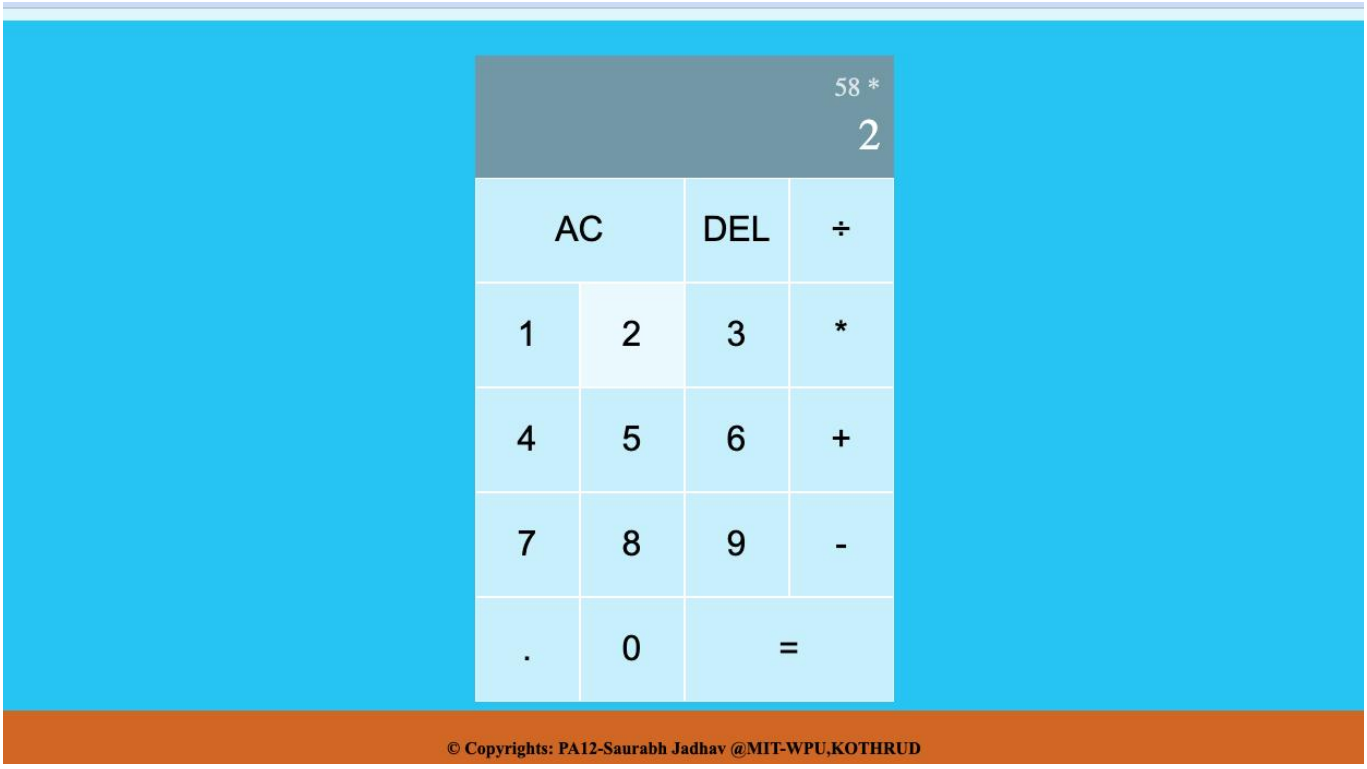
- **useState:** Allows functional components to have state.
- **useEffect:** Performs side effects in functional components (e.g., data fetching, subscriptions, manual DOM manipulations) and replaces lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount.
- **useContext:** Consumes values from React context.
- **useReducer:** Manages more complex state logic by using a reducer function.
- **useCallback and useMemo:** Memoize functions and values to optimize performance.

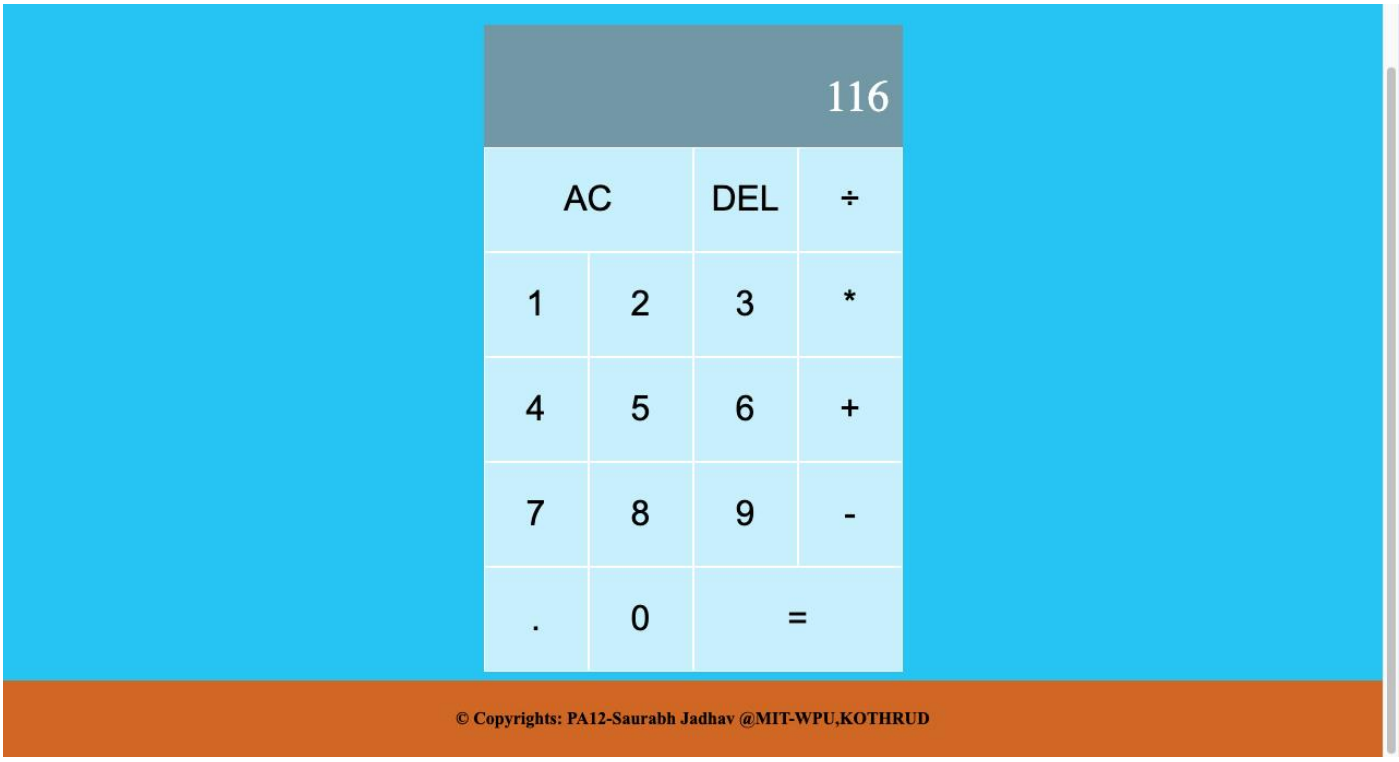
Implementation Screenshots:-





Working:-





Github Link:-

https://github.com/Saurabh3207/Calculator-React