Name: Tejas Redkar
PRN: 1032210937
Roll No.: 44
Panel: C, Batch: C2

FSD Lab 05

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 a React app using create-react-app:

React is a popular JavaScript library for building user interfaces, particularly single-page applications. It was developed by Facebook and is widely used for creating interactive and dynamic web applications. React allows developers to build reusable UI components and efficiently manage the state of the application.

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

Step 1: Install Node.js and npm:

Before you can create and run a React app, you need to have Node.js and npm (Node Package Manager) installed on your system. You can download and install them from [the official Node.js website](https://nodejs.org/).

Step 2: Install create-react-app (if not already installed):
You can install create-react-app globally using npm with the following

command:

```
npm install -g create-react-app
```

Step 3: Create a new React app:
To create a new React app, open your terminal and run the following

command:

```
npx create-react-app my-react-app
```

Replace my-react-app with your preferred app name.

Step 4: Change directory to your app:
Use the cd command to change your working directory to the newly created

app:

```
cd my-react-app
```

Step 5: Run the React app:
Once you are inside your app's directory, you can start the development

server with the following command:

```
npm start
```

This command will start the development server and open your React app in your default web browser. You can now begin building your React app by editing the code in the src directory.

2. Passing data through props (Small example):

In React, you can pass data from a parent component to a child component using props (short for properties). Props are a way to communicate and share data between React components. Here's a

small example of how you can pass data from a parent component to a child component:

ParentComponent.js:

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

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

return ( <div>

<h1>Parent Component</h1>

<ChildComponent message={data} /> </div>

); }

export default ParentComponent;
```

ChildComponent.js:

```
import React from 'react';

function ChildComponent(props) { return (

<div>
<h2>Child Component</h2> <p>{props.message}</p>

</div> );

}
export default ChildComponent;
```

In this example, the ParentComponent passes a message (the data variable) to the ChildComponent as a prop named message. The ChildComponent receives this prop and displays it within a paragraph

element. When you render the ParentComponent, you will see the message from the parent displayed within the child component.

FAQs:
1. What are the advantages of Server-side Scripting?

ANS:

React States: States are used to manage data that can change over time in React components, primarily in class components. They make components dynamic and responsive.

React Hooks: Hooks are functions that allow functional components to manage state and perform side effects, enabling more flexibility and functionality previously associated with class components.

OUTPUT Screenshots:

Calculator.js

```jsx
import React, { useState } from 'react';
import CalculatorButtons from './CalculatorButtons';
import './Calculator.css';

const Calculator = () => {
  const [input, setInput] = useState('');

  const handleButtonClick = (value) => {
    setInput((prevInput) => prevInput + value);
  };

  const handleEqualClick = () => {
    try {
      setInput(eval(input).toString());
    } catch (error) {
      setInput('Error');
    }
  };

  const handleClearClick = () => {
    setInput('');
  };

  return (
    <div className="calculator-container">
      <input type="text" value={input} readOnly />
      <CalculatorButtons
        onButtonClick={handleButtonClick}
        onEqualClick={handleEqualClick}
        onClearClick={handleClearClick}
      />
    </div>
  );
};

export default Calculator;
```

App.css

```css
.app-container {
  text-align: center;
  margin-top: 50px;
}

h1 {
  color: #3498db;
  font-size: 24px;
}

.sub-text {
  color: #3498db;
  font-size: 18px;
  font-weight: bold;
}
```

Calculator.css

```css
.calculator-container {
    display: inline-block;
    text-align: center;
    border: 1px solid #ccc;
    padding: 10px;
    margin-top: 20px;
  }

  .calculator-container input {
    width: 80%;
    margin-bottom: 10px;
    padding: 8px;
  }

```

CalculatorButtons.css

```css
.buttons-container {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-gap: 10px;
  }

  .buttons-container button {
    padding: 10px;
    font-size: 18px;
  }

```

CalculatorButtons.js

```jsx
import React from 'react';
import './CalculatorButtons.css';

const CalculatorButtons = ({ onButtonClick, onEqualClick, onClearClick }) => {
  const buttons = [
    '7', '8', '9', '/',
    '4', '5', '6', '*',
    '1', '2', '3', '-',
    '0', '.', '=', '+'
  ];

  return (
    <div className="buttons-container">
      {buttons.map((button, index) => (
        <button key={index} onClick={() => {
          if (button === '=') {
            onEqualClick();
          } else if (button === 'C') {
            onClearClick();
          } else {
            onButtonClick(button);
          }
        }}>
          {button}
        </button>
      ))}
    </div>
  );
};

export default CalculatorButtons;
```

App.js

```jsx
import React from 'react';
import Calculator from './Calculator';
import './App.css';

const App = () => {
  return (
    <div className="app-container">
      <h1>Design and develop a UI for calculator using React</h1>
      <p className="sub-text"><strong> PC Devanshu Surana  </strong></p>
      <Calculator />
    </div>
  );
};

export default App;
```

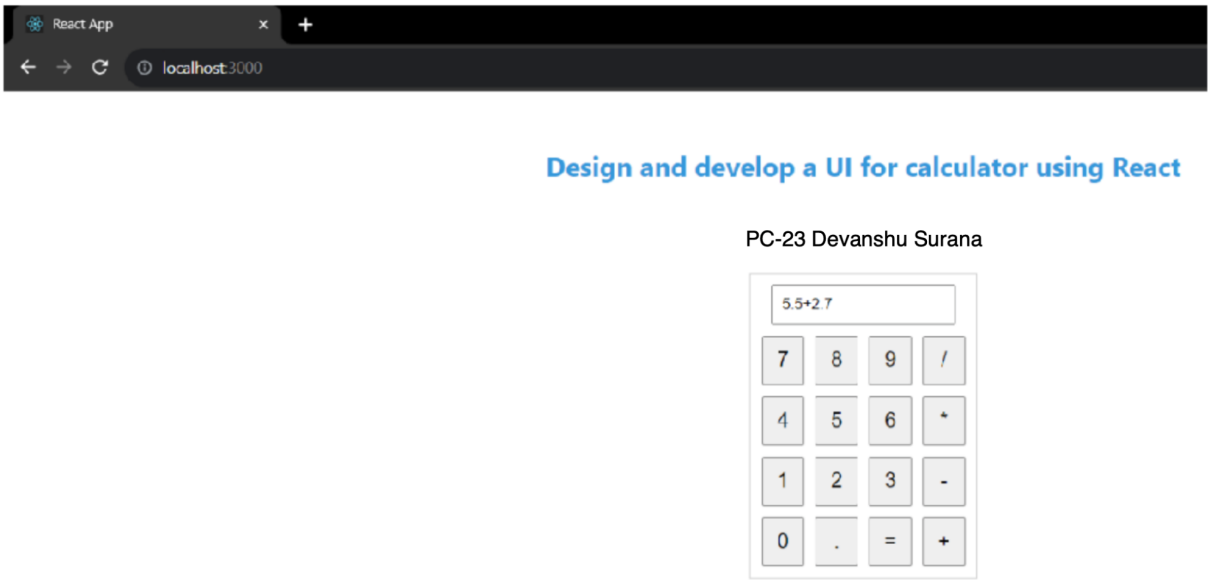
UI for calculator using React

Design and develop a UI for calculator using React

PC-23 Devanshu Surana

Mathematical operations

Inserting values:



OUTPUT: