**Name: Om**

**Panel: F**

**Batch: F1**

**Roll\_no: 16**

**Faculty: Prof. Vitthal Gutte**

**Subject: Full Stack Development**

**Title: Assignment-5**

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 an open-source JavaScript library used for building user interfaces or UI components for web applications. It allows developers to create interactive and dynamic web applications with ease.

- To create a React app using `create-react-app`, you can follow these steps:

- Install Node.js and npm (Node Package Manager) if you haven't already.

- Open your terminal and run the following command to create a new React app:

```

npx create-react-app my-app

```

- Replace `my-app` with your desired app name.

- Navigate to the newly created app directory:

```

cd my-app

```

- Start the development server:

```

npm start

```

- This will launch your React app in a development environment, and you can begin building your user interface.

2. Passing data through props :

- In React, you can pass data from a parent component to a child component using "props" (short for properties). Props allow you to send data and configuration to child components.

- Here's a simple example of passing data through props:

```jsx

// ParentComponent.js

import React from 'react';

import ChildComponent from './ChildComponent';

function ParentComponent() {

const data = 'Hello from Parent Component';

return (

<div>

<h2>Parent Component</h2>

<ChildComponent message={data} />

</div>

);

}

export default ParentComponent;

```

```jsx

// ChildComponent.js

import React from 'react';

function ChildComponent(props) {

return (

<div>

<h3>Child Component</h3>

<p>{props.message}</p>

</div>

);

}

export default ChildComponent;

```

- In this example, we have a `ParentComponent` passing the `message` prop to the `ChildComponent`. The child component then displays the received data using `{props.message}`.

**#FAQ:**

1. What are React states and hooks?

- React States: States in React are used to manage and store data that can change over time. Components can have their own state, and when the state changes, React re-renders the component. You can use the `useState` hook to manage state in functional components, or `this.state` and `this.setState` in class components.

- React Hooks: React introduced hooks in version 16.8 to allow functional components to have state, lifecycle methods, and more. Commonly used hooks include:

- `useState`: Manages state in functional components.

- `useEffect`: Handles side effects (e.g., data fetching) in functional components.

- `useContext`: Provides a way to access context in functional components.

- `useRef`: Allows access to the DOM or React elements.

- `useReducer`: Manages more complex state logic in functional components.

- and more.

Hooks enable functional components to do everything that class components can do, making it easier to manage and share logic in your React applications.

CODE

1. App.js
2. import Calculator from "./component/calculator";
3. import "./App.css";
4. function App() {
5. return (
6. <div>
7. <h1 style={{ textAlign: "center" }}>Aryan bansal Panel-F FSD\_5</h1>;
8. <h2 style={{ textAlign: "center" }}>Basic Calculator using ReactJS</h2>
9. ;
10. <Calculator />;
11. </div>
12. );
13. }
14. export default App;

2. Calculator.js

import React, { useState } from "react";

import "./calculator.css";

import Button from "./Button.jsx";

function Calculator() {

  const [input, setInput] = useState("");

  const handleClick = (value) => {

    if (value === "=") {

      setInput(eval(input).toString());

    } else if (value === "AC") {

      setInput("");

    } else if (value === "Del") {

      setInput(input.slice(0, -1));

    } else {

      setInput((prevState) => prevState + value);

    }

  };

  const buttonlist = [

    "7",

    "8",

    "9",

    "/",

    "4",

    "5",

    "6",

    "\*",

    "1",

    "2",

    "3",

    "-",

    "0",

    "AC",

    "=",

    "+",

    ".",

    "Del",

  ];

  return (

    <div className="calculator">

      <div className="displaypanel">{input}</div>

      <div className="buttons">

        {buttonlist.map((label) => (

          <Button key={label} onClick={handleClick} label={label} />

        ))}

      </div>

    </div>

  );

}

export default Calculator;

1. Calculator.css
2. .calculator {
3. width: 400px;
4. margin: 0 auto;
5. padding: 25px;
6. border: 1px solid #ccc;
7. border-radius: 10px;
8. box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
9. background-color: black;
10. }
11. .displaypanel {
12. font-size: 30px;
13. text-align: right;
14. margin-bottom: 10px;
15. padding: 15px;
16. background-color: #f7f7f7;
17. border: 1px solid #ccc;
18. border-radius: 5px;
19. height: 30px;
20. }
21. .buttons {
22. display: grid;
23. grid-template-columns: repeat(4, 1fr);
24. grid-gap: 10px;
25. }
26. button {
27. font-size: 18px;
28. padding: 10px;
29. border: 1px solid #ccc;
30. border-radius: 5px;
31. background-color: aqua;
32. cursor: pointer;
33. transition: background-color 0.3s ease;
34. &:hover {
35. background-color: #e0e0e0;
36. }
37. }
38. Button.jsx
39. import React from "react";
40. import "./calculator.css";
41. const Button = ({ label, onClick }) => {
42. return <button onClick={() => onClick(label)}>{label}</button>;
43. };
44. export default Button;

OUTPUT









