Creating a simple calculator in React is a great way to practice handling state, events, and conditional logic. Below is a basic example of a calculator with functionality for addition, subtraction, multiplication, and division.

Calculator Project

Project Structure

- src
  - components
    - Calculator.js
  - App.js
  - index.js
  - App.css

Calculator.js

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

function Calculator() {
  const [input, setInput] = useState('');
  const [result, setResult] = useState('');

  const handleClick = (value) => {
    setInput(input + value);
  };

  const handleClear = () => {
    setInput('');
    setResult('');
  };

  const handleCalculate = () => {
    try {
      setResult(eval(input).toString());
    } catch {
      setResult('Error');
    }
  };
```

```
  return (
    <div className="calculator">
      <div className="display">
        <div className="input">{input}</div>
        <div className="result">{result}</div>
      </div>
      <div className="buttons">
        <button onClick={() => handleClick('1')}>1</button>
        <button onClick={() => handleClick('2')}>2</button>
        <button onClick={() => handleClick('3')}>3</button>
        <button onClick={() => handleClick('+')}>+</button>
        <button onClick={() => handleClick('4')}>4</button>
        <button onClick={() => handleClick('5')}>5</button>
        <button onClick={() => handleClick('6')}>6</button>
        <button onClick={() => handleClick('-')}>-</button>
        <button onClick={() => handleClick('7')}>7</button>
        <button onClick={() => handleClick('8')}>8</button>
        <button onClick={() => handleClick('9')}>9</button>
        <button onClick={() => handleClick('*')}>*</button>
        <button onClick={() => handleClick('0')}>0</button>
        <button onClick={() => handleClick('.')}>.</button>
        <button onClick={handleCalculate}>=</button>
        <button onClick={() => handleClick('/')}>/</button>
        <button onClick={handleClear}>C</button>
      </div>
    </div>
  );
}

export default Calculator;
```

App.js

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

function App() {
  return (
    <div className="App">
```

```
      <Calculator />
    </div>
  );
}

export default App;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

App.css (for basic styling)

css

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #f5f5f5;
}

.calculator {
  border: 1px solid #ccc;
  border-radius: 4px;
  background-color: #fff;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
```

```
  width: 200px;
}

.display {
  background-color: #333;
  color: #fff;
  padding: 10px;
  text-align: right;
  border-top-left-radius: 4px;
  border-top-right-radius: 4px;
}

.input {
  font-size: 20px;
  overflow: hidden;
}

.result {
  font-size: 24px;
}

.buttons {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
}

button {
  border: 1px solid #ccc;
  background-color: #fff;
  padding: 20px;
  font-size: 18px;
  cursor: pointer;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #f0f0f0;
}

button:active {
  background-color: #ddd;
}
```

Explanation

1. **State Management**:
    - input: Stores the current input string (numbers and operators).
    - result: Stores the result of the calculation.
2. **Event Handlers**:
    - handleClick(value): Appends the clicked button's value to the input.
    - handleClear(): Resets both input and result states.
    - handleCalculate(): Evaluates the mathematical expression in input and updates the result. It uses eval()for simplicity, but in production, consider using a safer parsing library or custom parser.
3. **Display and Buttons**:
    - The calculator has a display area showing both the input and the result.
    - Buttons for digits, operators, and functions like clear (C) and equals (=).
4. **Styling**:
    - Basic CSS styles the calculator layout, display area, and buttons, providing a clean and user-friendly interface.

This simple calculator can be expanded with additional features like keyboard support, advanced mathematical functions, or improved styling.