

# Web Development Project Report

## Project Overview

### Project Name:

Calculator Application

### Description:

This web application is a calculator that allows users to perform basic arithmetic operations (addition, subtraction, multiplication, division) on numbers. It provides a user-friendly interface for entering digits and performing calculations.

## Technologies Used

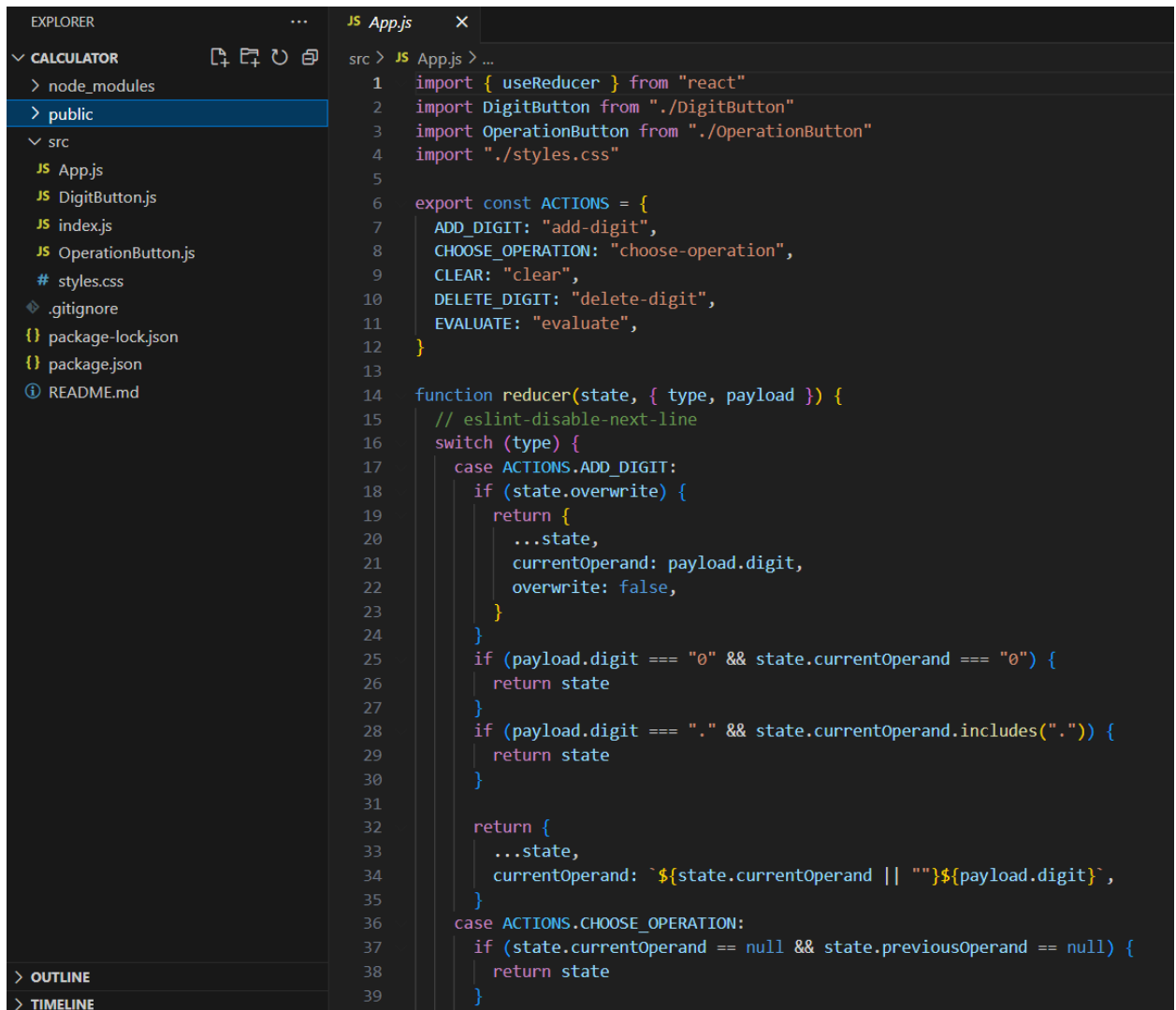
- Frontend Framework: React
- Styling: CSS (with some external styling libraries)
- Version Control: Git (with GitHub for repository hosting)
- Package Management: npm (Node Package Manager)
- Code Editor: Visual Studio Code

## Project Structure

The project is structured as follows:

- ``src/`` (Source Code)

- `App.js`: Main component for the calculator application.



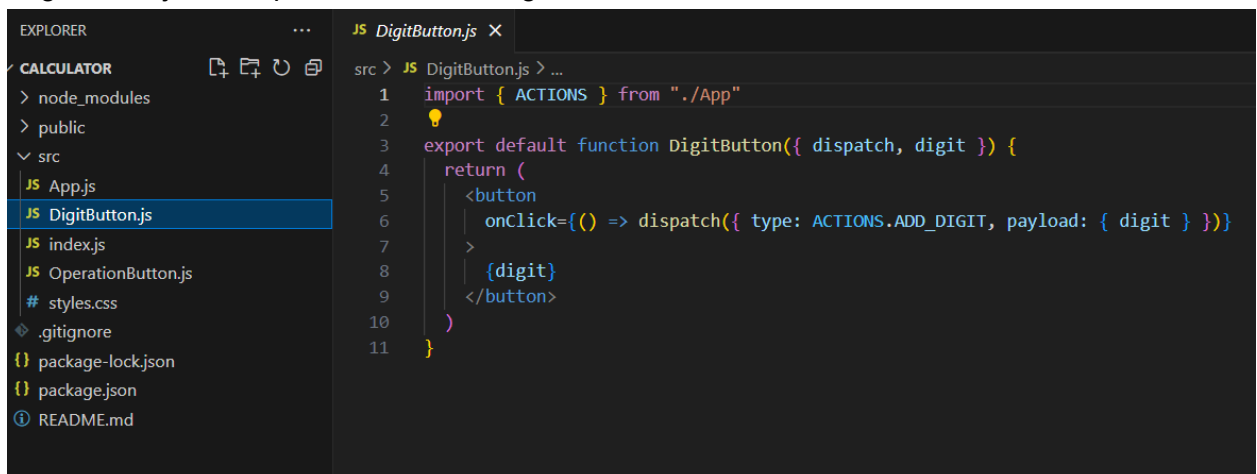
```

EXPLORER
  CALCULATOR
    > node_modules
    > public
    > src
      JS App.js
      JS DigitButton.js
      JS index.js
      JS OperationButton.js
      # styles.css
      .gitignore
      {} package-lock.json
      {} package.json
      README.md

src > JS App.js > ...
1  import { useReducer } from "react"
2  import DigitButton from "./DigitButton"
3  import OperationButton from "./OperationButton"
4  import "./styles.css"
5
6  export const ACTIONS = {
7    ADD_DIGIT: "add-digit",
8    CHOOSE_OPERATION: "choose-operation",
9    CLEAR: "clear",
10   DELETE_DIGIT: "delete-digit",
11   EVALUATE: "evaluate",
12 }
13
14 function reducer(state, { type, payload }) {
15   // eslint-disable-next-line
16   switch (type) {
17     case ACTIONS.ADD_DIGIT:
18       if (state.overwrite) {
19         return {
20           ...state,
21           currentOperand: payload.digit,
22           overwrite: false,
23         }
24       }
25       if (payload.digit === "0" && state.currentOperand === "0") {
26         return state
27       }
28       if (payload.digit === "." && state.currentOperand.includes(".")) {
29         return state
30       }
31
32       return {
33         ...state,
34         currentOperand: `${state.currentOperand || ""}${payload.digit}`,
35       }
36     case ACTIONS.CHOOSE_OPERATION:
37       if (state.currentOperand == null && state.previousOperand == null) {
38         return state
39       }

```

- `DigitButton.js`: Component for rendering numeric buttons.



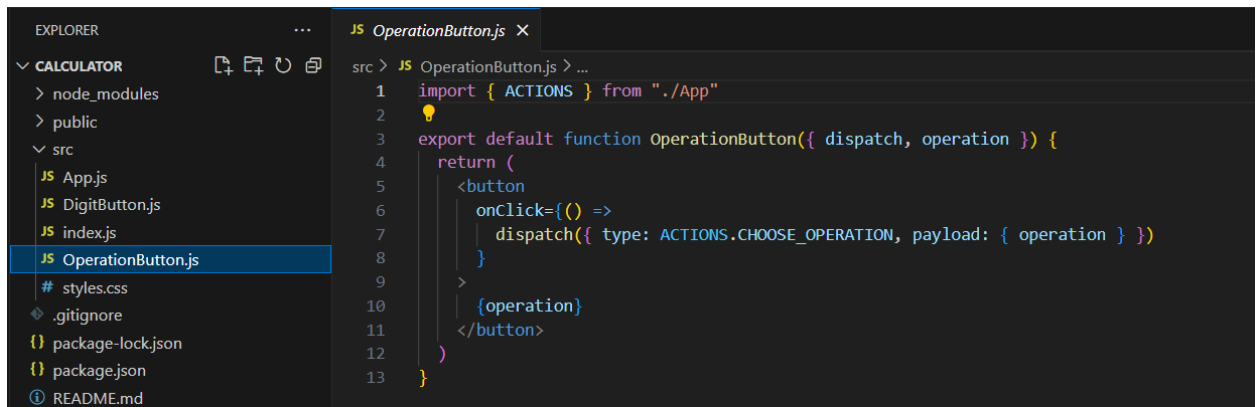
```

EXPLORER
  CALCULATOR
    > node_modules
    > public
    > src
      JS App.js
      JS DigitButton.js
      JS index.js
      JS OperationButton.js
      # styles.css
      .gitignore
      {} package-lock.json
      {} package.json
      README.md

src > JS DigitButton.js > ...
1  import { ACTIONS } from "./App"
2
3  export default function DigitButton({ dispatch, digit }) {
4    return (
5      <button
6        onClick={() => dispatch({ type: ACTIONS.ADD_DIGIT, payload: { digit } })}
7      >
8        {digit}
9      </button>
10    )
11  }

```

- `OperationButton.js`: Component for rendering operation buttons.



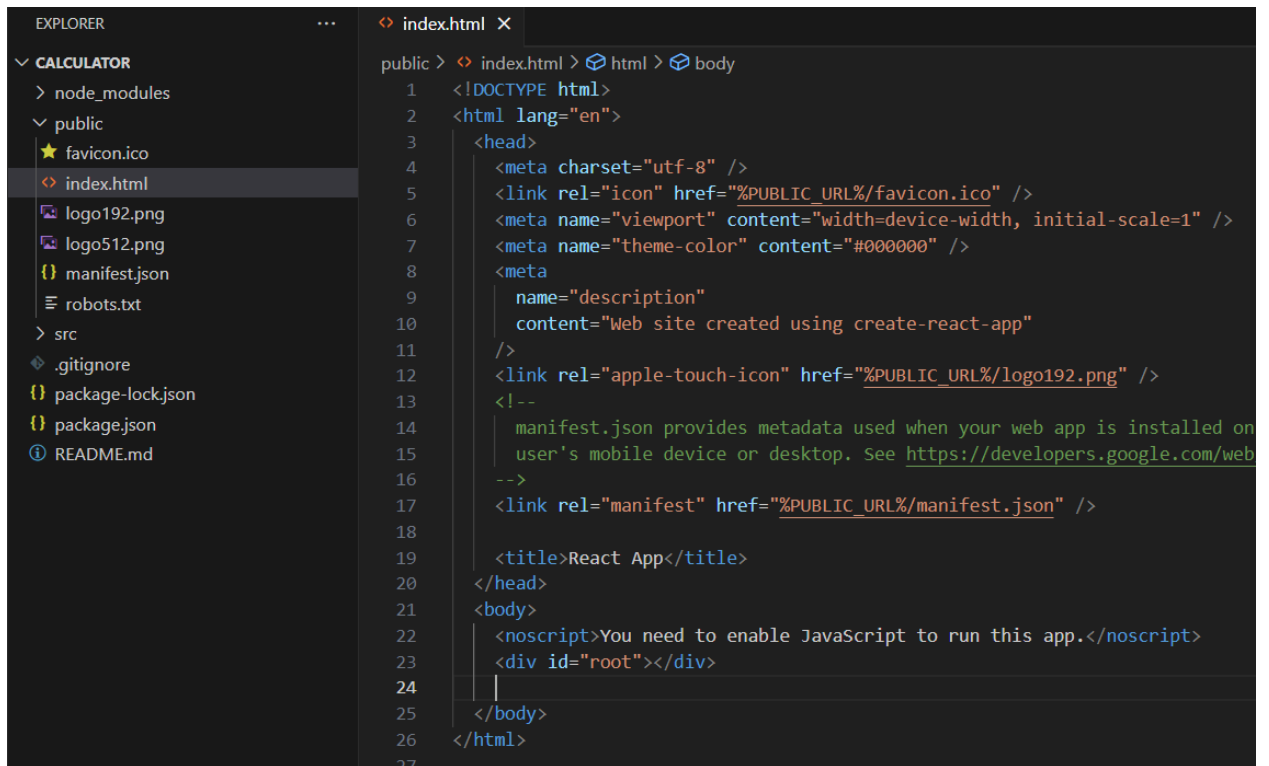
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'CALCULATOR' with a 'src' directory containing 'App.js', 'DigitButton.js', 'index.js', and 'OperationButton.js'. The 'OperationButton.js' file is selected. The code editor shows the following code:

```
src > JS OperationButton.js > ...
1  import { ACTIONS } from "../App"
2
3  export default function OperationButton({ dispatch, operation }) {
4    return (
5      <button
6        onClick={() =>
7          dispatch({ type: ACTIONS.CHOOSE_OPERATION, payload: { operation } })
8        }
9      >
10     {operation}
11   </button>
12 )
13 }
```

- `styles.css`: CSS file for styling the application.

```
1 *, ::before, ::after {
2   box-sizing: border-box;
3 }
4
5 body {
6   margin: 0;
7   background: linear-gradient(to right, #00AAFF, #00F
8 }
9
10 .calculator-grid {
11   display: grid;
12   margin-top: 2rem;
13   justify-content: center;
14   grid-template-columns: repeat(4, 6rem);
15   grid-template-rows: minmax(7rem, auto) repeat(5, 6rem);
16 }
17
18 .calculator-grid > button {
19   cursor: pointer;
20   font-size: 2rem;
21   border: 1px solid white;
22   outline: none;
23   background-color: rgba(255, 255, 255, .75);
24 }
25
26 .calculator-grid > button:hover,
27 .calculator-grid > button:focus {
28   background-color: rgba(255, 255, 255, .9);
29 }
30
31 .span-two {
32   grid-column: span 2;
33 }
34
35 .output {
36   grid-column: 1 / -1;
37   background-color: rgba(0, 0, 0, .75);
38   display: flex;
```

- `public/`: Contains the HTML file (`index.html`) and other static assets.



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'CALCULATOR'. The tree includes folders like 'node\_modules' and 'public', and files such as 'favicon.ico', 'index.html', 'logo192.png', 'logo512.png', 'manifest.json', 'robots.txt', 'src', '.gitignore', 'package-lock.json', 'package.json', and 'README.md'. The 'index.html' file is selected. The main editor area shows the content of 'index.html', which is an HTML document. The code includes a DOCTYPE declaration, an HTML lang attribute, a head section with meta tags for charset, viewport, theme-color, and a description, and link tags for an icon, an apple touch icon, and a manifest file. The body section contains a noscript message and a root div.

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7     <meta name="theme-color" content="#000000" />
8     <meta
9       name="description"
10      content="Web site created using create-react-app"
11    />
12    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13    <!--
14      manifest.json provides metadata used when your web app is installed on
15      user's mobile device or desktop. See https://developers.google.com/web
16    -->
17    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
18
19    <title>React App</title>
20  </head>
21  <body>
22    <noscript>You need to enable JavaScript to run this app.</noscript>
23    <div id="root"></div>
24
25  </body>
26 </html>
27

```

## Features

### 1. Basic Arithmetic Operations:

- Addition, subtraction, multiplication, and division operations are supported.

### 2. Clear and Delete Functionality:

- Users can clear the display or delete the last entered digit.

### 3. Decimal Point Handling:

- The application ensures that only one decimal point can be entered in a number.

### 4. Error Handling:

- The application is designed to handle edge cases and prevent invalid input.

### 5. State Management:

- State is managed using the `useReducer` hook, ensuring predictable and controlled updates.

## Code Structure

- Component-Based Architecture:

The application is built using a component-based architecture. Each component handles a specific aspect of the UI (e.g., buttons, display).

- **Reducer for State Management:**  
State is managed using a reducer function, which processes actions and returns a new state based on the action type.
- **Spread Operator for Immutability:**  
The spread operator (`...`) is used to create new state objects, ensuring immutability and predictable state updates.

## Future Improvements

1. **Scientific Calculator Features:** Addition of advanced functions (trigonometric, logarithmic, etc.).
2. **History of Calculations:** Implementing a history feature to track previous calculations.
3. **Improved Styling:** Enhancing the visual design for a more polished user interface.

## ScreenShot

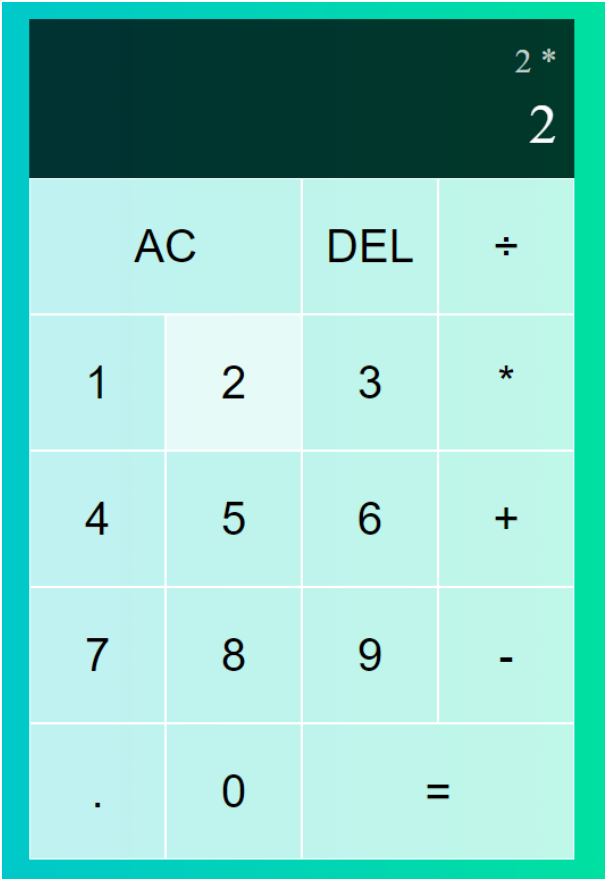
1.



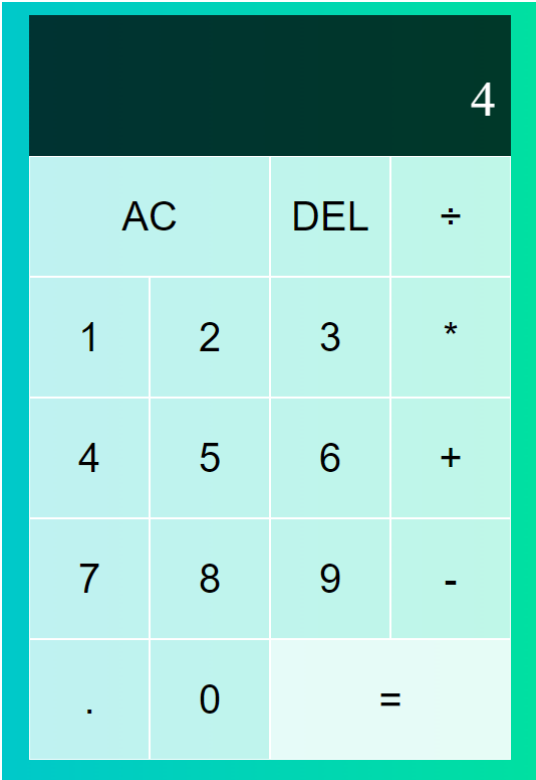
2

AC		DEL	÷
1	2	3	*
4	5	6	+
7	8	9	-
.	0	=	

2.



3.



4.



## Conclusion

The Calculator Application project demonstrates proficiency in React and modern web development practices. The use of component-based architecture, state management, and immutability techniques contributes to the maintainability and scalability of the application. Further enhancements and refinements can be made to elevate the user experience.