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
    App.js
    DigitButton.js
    index.js
    OperationButton.js
    styles.css
  .gitignore
  package-lock.json
  package.json
  README.md
OUTLINE
TIMELINE
```

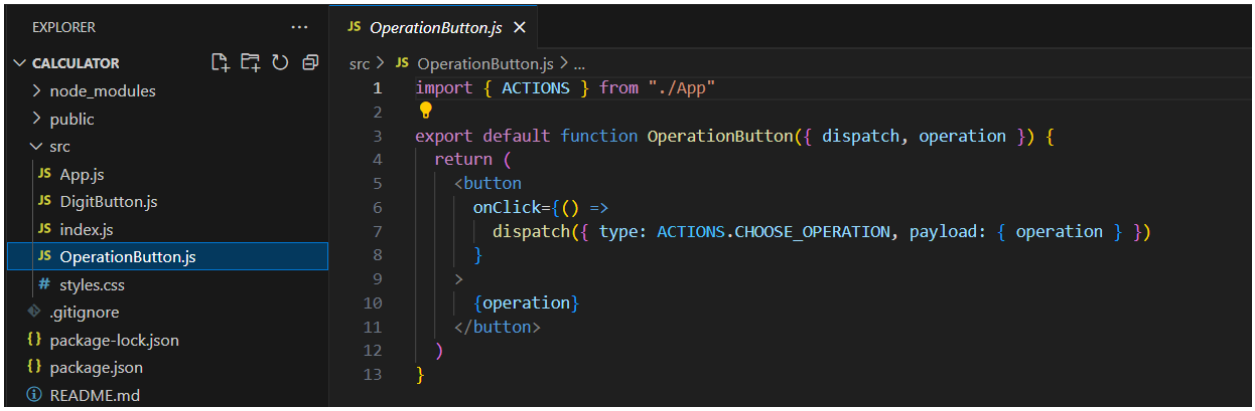
```
App.js
src > App.js > ...
import { useReducer } from "react"
import DigitButton from "./DigitButton"
import OperationButton from "./OperationButton"
import "./styles.css"

export const ACTIONS = {
  ADD_DIGIT: "add-digit",
  CHOOSE_OPERATION: "choose-operation",
  CLEAR: "clear",
  DELETE_DIGIT: "delete-digit",
  EVALUATE: "evaluate",
}

function reducer(state, { type, payload }) {
  // eslint-disable-next-line
  switch (type) {
    case ACTIONS.ADD_DIGIT:
      if (state.overwrite) {
        return {
          ...state,
          currentOperand: payload.digit,
          overwrite: false,
        }
      }
      if (payload.digit === "0" && state.currentOperand === "0") {
        return state
      }
      if (payload.digit === "." && state.currentOperand.includes(".")) {
        return state
      }

      return {
        ...state,
        currentOperand: `${state.currentOperand || ""}${payload.digit}`,
      }
    case ACTIONS.CHOOSE_OPERATION:
      if (state.currentOperand == null && state.previousOperand == null) {
        return state
      }
```

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

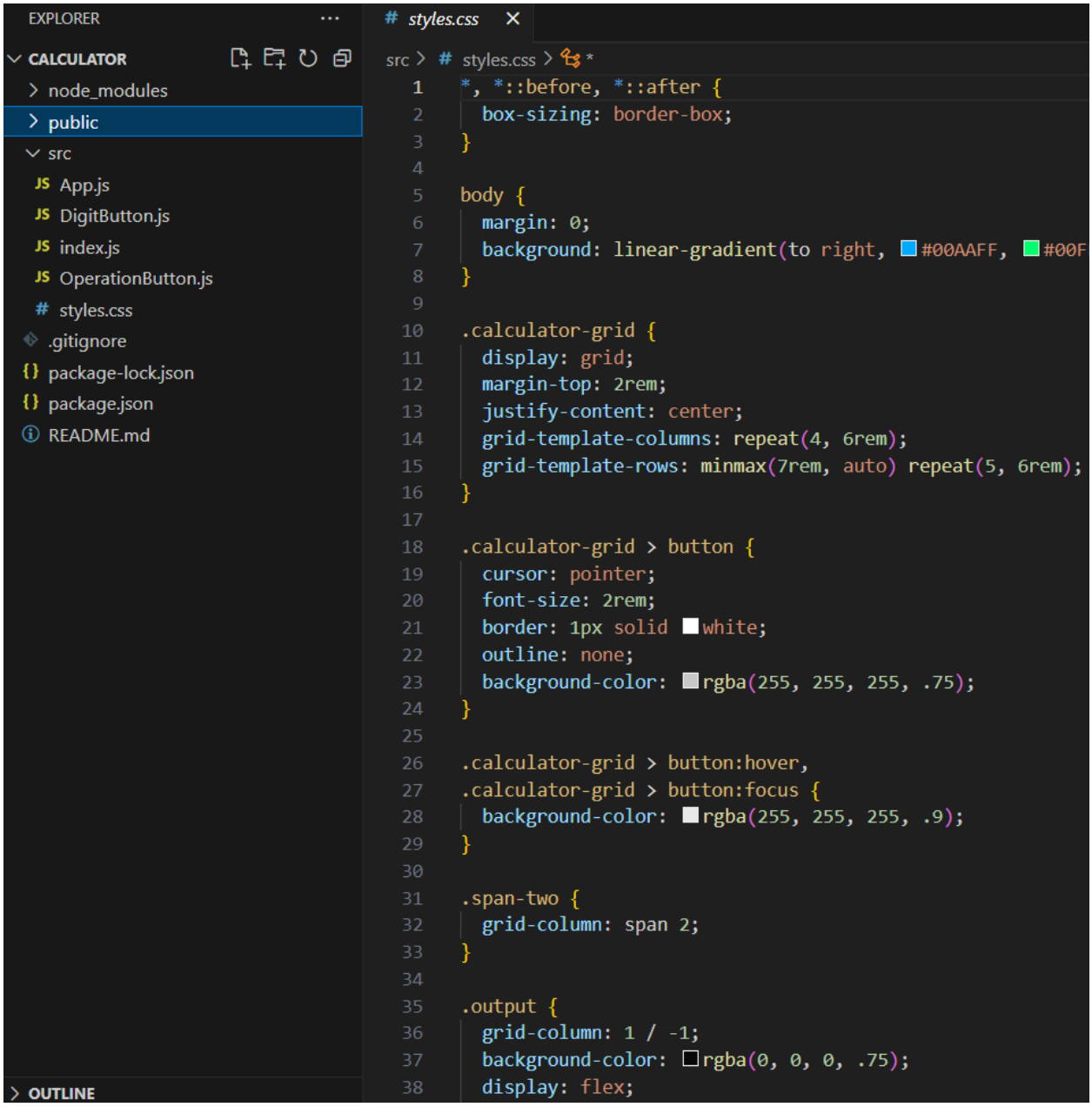
```
EXPLORER
CALCULATOR
  node_modules
  public
  src
    App.js
    DigitButton.js
    index.js
    OperationButton.js
    styles.css
  .gitignore
  package-lock.json
  package.json
  README.md
```

```
DigitButton.js
src > DigitButton.js > ...
import { ACTIONS } from "./App"

export default function DigitButton({ dispatch, digit }) {
  return (
    <button
      onClick={() => dispatch({ type: ACTIONS.ADD_DIGIT, payload: { digit } })}
    >
      {digit}
    </button>
  )
}
```

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

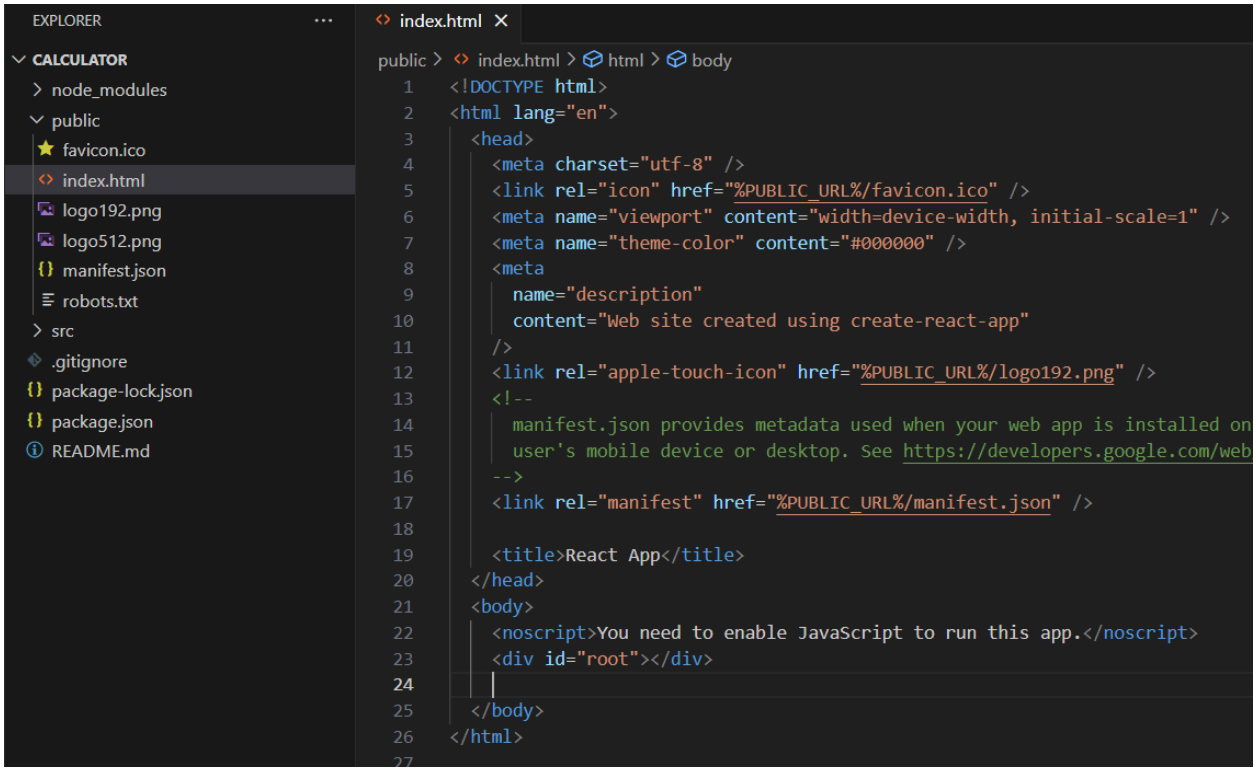


- `styles.css`: CSS file for styling the application.




```
*, *::before, *::after {
  box-sizing: border-box;
}

body {
  margin: 0;
  background: linear-gradient(to right, #00AAFF, #00F
}

.calculator-grid {
  display: grid;
  margin-top: 2rem;
  justify-content: center;
  grid-template-columns: repeat(4, 6rem);
  grid-template-rows: minmax(7rem, auto) repeat(5, 6rem);
}

.calculator-grid > button {
  cursor: pointer;
  font-size: 2rem;
  border: 1px solid white;
  outline: none;
  background-color: rgba(255, 255, 255, .75);
}

.calculator-grid > button:hover,
.calculator-grid > button:focus {
  background-color: rgba(255, 255, 255, .9);
}

.span-two {
  grid-column: span 2;
}

.output {
  grid-column: 1 / -1;
  background-color: rgba(0, 0, 0, .75);
  display: flex;
```

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



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.).

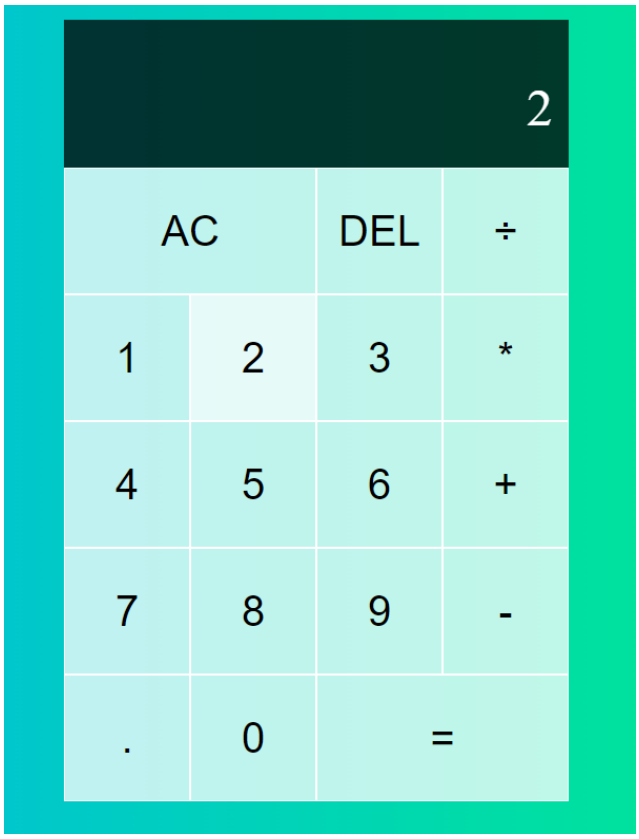
2.History of Calculations: Implementing a history feature to track previous calculations.

3. Improved Styling: Enhancing the visual design for a more polished user interface.

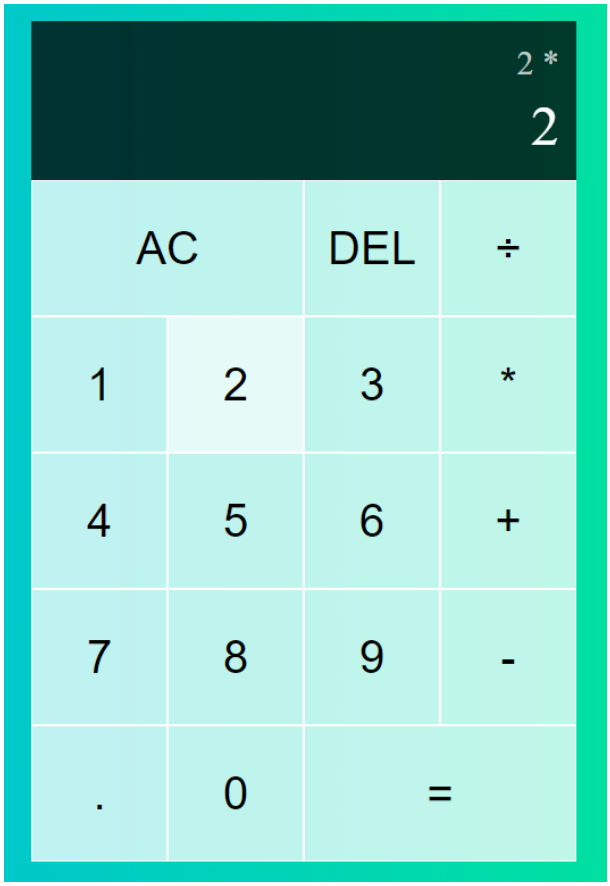
ScreenShot

1. 

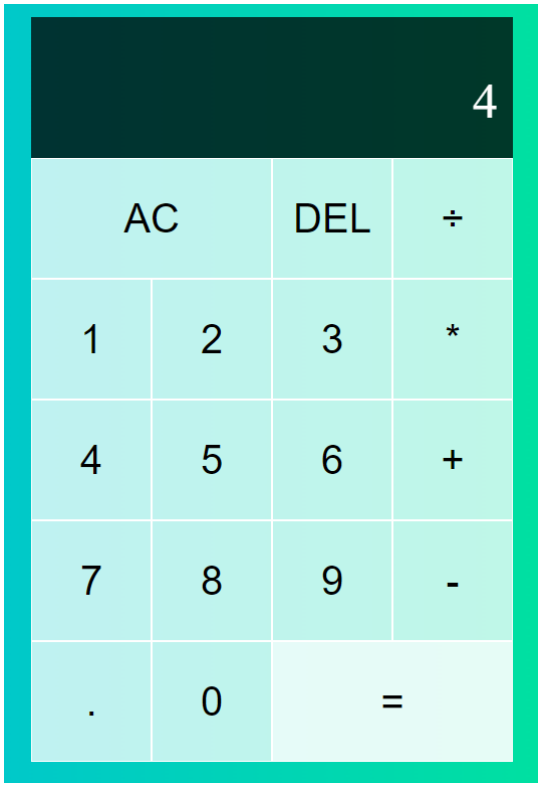
2.



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.