Subject Name: **Front End Engineering**

Subject Code: CS186

Cluster: iGamma Group: **G19**

Department: **DCSE**

Project Name: Calculator App

Submitted By:

Manav Shandilya

2110992084

G19

Submitted To:

Mr. Rishik

Node.js Calculator App

Introduction

Welcome to the documentation of our React-based Note-Keeping App. In this documentation, we will explore the key components that make up our application and their respective functionalities.

The Node.js Calculator App is a web-based application that provides basic calculations. It allows users to perform addition, subtraction, multiplication, and division operations. The application is built using Node.js for the server-side logic and Express.js as the web framework.

Technologies Used:

- HTML
- CSS
- JavaScript
- React js

Components

1. App Component

- The central component of our application that manages the state of notes.

- Allows users to add and delete notes.

- Coordinates the interactions between other components.

2. Digit Button Component

- The digit component is responsible for capturing user input when a digit button is clicked on the calculator interface.

- It manages the display of entered digits on the calculator screen, allowing users to see the numbers they input.

- Validates user input to ensure that only valid digits are accepted. This may include checking for duplicate decimal points, preventing leading zeros, etc.
- Supports functionality to clear the entered digits, allowing users to start a new calculation.

- The digit component communicates with the calculator logic to update the internal state of the calculation based on the entered digits.

3. Operation Button Component

- The operation button component is responsible for capturing user input when an operation button (addition, subtraction, multiplication, division) is clicked on the calculator interface.

- Manages the internal state of the calculator, indicating the current operation to be performed when the equal button is pressed.

- Communicates with the calculator logic to update the internal state of the calculation based on the selected operation.

Usage:

Calculator App comprises server-side components (such as app.js and routing modules) and front-end components (including styling, client-side logic, and EJS templates). User interactions are managed by the digit, operation button, and equals button components. The digit component captures and displays numeric input, the operation button component handles arithmetic operations, and the equals button component finalizes and displays calculation results. Additional components like utility functions, images, and documentation contribute to a modular and maintainable architecture. The application provides a responsive and intuitive interface for performing basic arithmetic calculations.

In the context of a calculator application built with Node.js and Express.js, various components work together to provide a functional and interactive user experience. Let's discuss the usage of each component in the context of user interactions:

1. App.js

Usage:

- Initializes the Express.js application.
- Sets up middleware for handling static files, such as CSS and JavaScript.
- Defines routes and their corresponding handlers.

2. styles.css

Usage:

- Defines the visual appearance of HTML elements.
- Ensures a visually appealing and responsive design for the calculator.

3. script.js

Usage:

- Captures user interactions, such as button clicks.
- Communicates with the digit and operation components based on user input.
- Updates the display to reflect user input and calculation results.

4. Digit Component

Usage:

- Captures and processes digit button clicks.
- Manages the display of entered digits on the calculator screen.
- Coordinates with the operation component to update the ongoing calculation.
- Supports clearing entered digits and resetting the state for a new calculation.

5. Operation Button Component

Usage:

- Captures and processes operation button clicks (addition, subtraction, multiplication, division).
- Coordinates with the digit component to set the ongoing operation.
- Manages the internal state of the calculator to indicate the current operation.
- Updates the display to reflect the selected operation.

6. README.md

Usage:

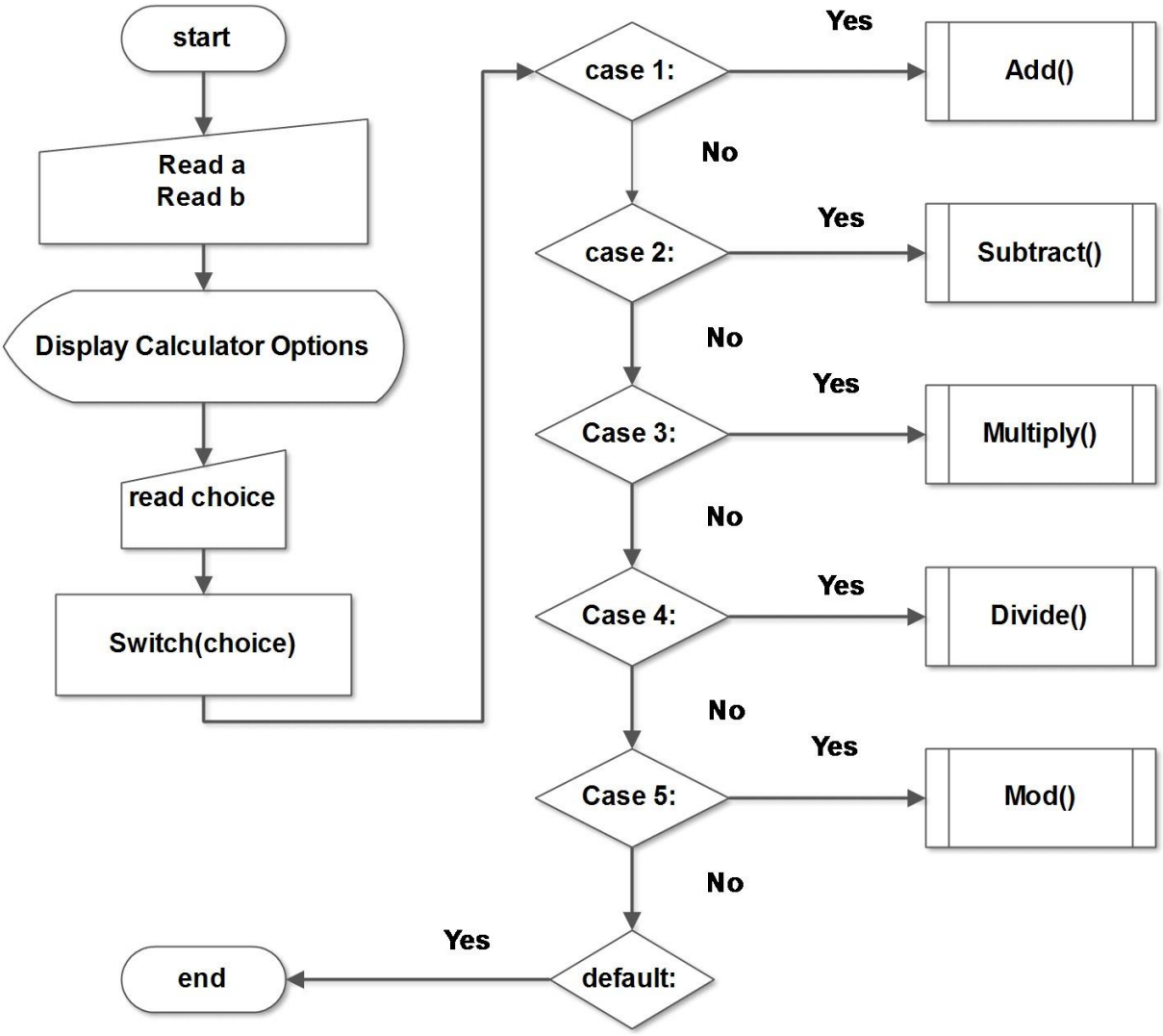
- Provides essential information about the project, including installation instructions and usage guidelines.

7. package.json

Usage:

- Includes metadata about the project, dependencies, and scripts for running the application.

Flow Chart:



----------Components Documentation----------

Index.js

import React from "react"; and import ReactDOM from "react-dom";:

Imports the React library for building user interfaces and the ReactDOM library for rendering React components into the DOM. import App from "./App";:

Imports the main component of the application, assumed to be named "App". The path indicates that the "App" component is in the same directory as the file or in a directory named "App." ReactDOM.render(...);:

Renders the main component (<App />) into the root HTML element of the web page. <React.StrictMode>:

Wraps the App component with React.StrictMode. This is a development mode feature that helps detect common mistakes in the application by highlighting them or triggering warnings. document.getElementById("root"):

Identifies the HTML element with the id "root" in the HTML file where the React application will be injected. This is a common convention in React applications.

This file typically serves as the entry point for your React application, and the actual application logic and UI structure are defined in the App component and its child components.

App.js

The App component is the main component of the React Calculator App.

Utilizes React Hooks (useReducer) for state management.

Imports the DigitButton and OperationButton components.

Imports the global styles defined in styles.css.

Uses useReducer to manage the state of the calculator.
The reducer function handles various actions triggered by button clicks.

Enumerated actions in the ACTIONS object represent different user interactions.

Actions include adding digits, choosing operations, clearing, deleting digits, and evaluating expressions.

DigitButton.js

The DigitButton component represents a button for numeric digits (0-9) and the decimal point.

dispatch (function): The dispatch function to update the calculator state.

digit (string): The numeric value represented by the button.

The DigitButton component handles the click event for the button.
It dispatches the ADD_DIGIT action with the payload containing the clicked digit to update the calculator state.

The handleClick function is invoked when the button is clicked.

It dispatches the ADD_DIGIT action to update the calculator state with the clicked digit.

Import the DigitButton component into the desired parent component.

Include it within the JSX structure of the parent component, providing the required dispatch and digit props.

OperationButton.js

The OperationButton component represents a button for arithmetic operations (+, -, *, ÷).

dispatch (function): The dispatch function to update the calculator state.

operation (string): The arithmetic operation symbol.

The OperationButton component handles the click event for the button.

It dispatches the CHOOSE_OPERATION action with the payload containing the clicked operation symbol to update the calculator state.

The handleClick function is invoked when the button is clicked.

It dispatches the CHOOSE_OPERATION action to update the calculator state with the clicked operation symbol.

Import the OperationButton component into the desired parent component.

Include it within the JSX structure of the parent component, providing the required dispatch and operation props.