### **Folder Structure Explanation:**

* **public/**: Contains static assets and the main index.html file.
* **src/**: Contains the source code for the application.
  + **components/**: Directory for reusable components.
    - **Button.js**: Component for calculator buttons.
    - **Display.js**: Component for displaying calculator input and output.
    - **Calculator.js**: Main calculator component that integrates Button and Display.
  + **App.js**: Main component that renders the Calculator component.
  + **App.css**: Single CSS file for styling the entire application.
  + **index.js**: Entry point for React rendering.
* **.gitignore**: Git ignore file to specify which files and directories to ignore in version control.
* **package.json**: Configuration file for npm/yarn that includes dependencies, scripts, and other metadata.
* **README.md**: README file with instructions on how to use, develop, and deploy the calculator application.

### **Approach:**

* **CSS Organization**: All styles are contained within App.css, which is imported into App.js to style the entire application. We can organize styles within App.css using selectors specific to each component (Button, Display, Calculator).
* **Component Separation**: Each component (Button, Display, Calculator) remains focused on its specific functionality, ensuring maintainability and reusability.

### **Getting Started:**

1. **Create React App**: Initialized a new React project using create-react-app.
2. **Structure Components**: Created components (Button.js, Display.js, Calculator.js) .
3. **Style with App.css**: Utilized App.css to style components uniformly across the application.

### 

### 

### 

### **1. Display Component**

**Purpose:** The Display component is responsible for rendering and displaying the input and output of the calculator.

**Key Features:**

* Displays the current input expression or result.
* Handles edge cases like displaying errors ('Error') when calculations fail.

**Implementation Details:**

* Used React's state management to update the displayed value (input or result).

### **2. Button Component**

**Purpose:** The Button component represents each interactive button on the calculator interface.

**Key Features:**

* Renders a clickable button with a label corresponding to its functionality.
* Handles user interactions such as clicks, triggering specific calculator operations.

**Implementation Details:**

* Receives label as a prop to dynamically render different buttons.
* Executes a callback (onClick) when the button is clicked, triggering relevant operations in the calculator.

### **3. Calculator Component**

**Purpose:** The Calculator component integrates the Display and Button components to create the entire calculator interface and functionality.

**Key Features:**

* Manages the calculator's state including current input (input), calculation result (result), memory operations, and angle unit (angleUnit).
* Executes arithmetic, trigonometric, exponential, and other mathematical operations based on user input.
* Utilizes mathjs library for advanced mathematical calculations (sqrt, pow, sin, cos, etc.).

**Implementation Details:**

* Uses React's useState for managing component-level state.
* Implements event handlers (handleButtonClick, calculateResult) to respond to user interactions and update the calculator's display.
* Provides a cohesive user experience by integrating with Button and Display components.