MLSA INTERNSHIP

# introduction

NAME *–* ***Pravesh Keshri***

BRANCH/SECTION *-* *CS-3C*

ROLL NO. *-* *2300290120167*

PROJECT *–* ***create a simple calculator***

- TECH STACK

1 **HTML**: Structure the calculator's user interface, including buttons, display area, and layout.

2 **CSS**: Style the calculator for a visually appealing user interface, including colors, fonts, and button designs.

3. **JavaScript**: Implement the calculator's functionality, handling user input, performing calculations, and updating the display.
4. **Frameworks/Libraries (optional)**: Use frameworks like React or Vue.js for a more dynamic and organized structure, particularly for larger projects.
5. **Version Control (Git)**: Manage code versions and collaborate with others using Git for tracking changes.
6. **Development Environment**: Set up a local development environment using tools like Visual Studio Code or any IDE of choice for coding and testing.

- FEATURES

1. Basic Operations: Implement functions for addition, subtraction, multiplication, and division.

2. Clear and Clear Entry (C/CE): Include buttons to clear the entire input or just the last entry, allowing users to easily correct mistakes.

3. **User-Friendly Interface**: Design an intuitive layout with clearly labeled buttons for ease of use.

4. **Display Screen**: Provide a display area to show the current input and the results of calculations.

5. **Keyboard Support**: Allow users to perform calculations using both the on-screen buttons and keyboard inputs for convenience.

6. **Error Handling**: Include mechanisms to handle errors, such as division by zero or invalid input, and display appropriate messages.

# FOLDER STRUCTURE

- public/ : assests

- src/

/src: Main source folder containing all application code.

- **/components**: Contains reusable components (e.g., buttons, display).
- **/styles**: CSS files or style components for the application's styling.

/public: Contains static files that can be served directly, such as HTML files and images.

- **index.html**: The main HTML file where the app is mounted.

/assets: (Optional) Folder for images, icons, or other media files used in the calculator.

/utils: Utility functions for calculations, such as handling operations or input validation.

/tests: Contains test files to ensure the functionality of the calculator components and logic.

- LEARNING OUTCOMES

1. Understanding Basic Programming Concepts: Gain familiarity with fundamental programming concepts such as variables, functions, loops, and conditionals.

2. Proficiency in Web Development Technologies: Develop skills in HTML, CSS, and JavaScript, applying them to create a functional user interface.

3. Problem-Solving Skills: Enhance problem-solving abilities by tackling challenges related to user input, error handling, and calculation logic.

4. UI/UX Design Principles: Learn about basic user interface and user experience design principles, focusing on creating an intuitive and visually appealing layout.

5. Version Control Familiarity: Understand the use of version control systems (like Git) for tracking changes and collaborating on code.

6. Debugging Techniques: Improve debugging skills by identifying and resolving issues in code, ensuring the calculator functions correctly.

- FUTURE SCOPE

1. Advanced Functions: Integrate more complex mathematical functions, such as square roots, exponents, trigonometric functions, and logarithms.

2. Graphing Capabilities: Add a feature to graph mathematical equations, providing a visual representation of functions.

3. History Feature: Implement a history log to keep track of previous calculations for user reference.

4. Unit Conversions: Include functionality for converting between different units of measurement (e.g., length, weight, temperature).

5. Mobile and Desktop Apps: Expand the project into native mobile (iOS/Android) or desktop applications using frameworks like React Native or Electron.

6. Customizable Themes: Allow users to choose from various themes or color schemes to personalize their calculator experience.

# backend schema

User Schema:

- **userId**: Unique identifier for the user (e.g., UUID).
- **username**: String for the user's name.
- **passwordHash**: String for storing hashed passwords.
- **email**: String for user contact.

Calculation History Schema:

- **historyId**: Unique identifier for each calculation entry.
- **userId**: Reference to the user who made the calculation.
- **expression**: String representing the calculation (e.g., "2 + 2").
- **result**: The result of the calculation.
- **timestamp**: Date and time when the calculation was made.

Settings Schema:

- **userId**: Reference to the user.
- **theme**: String to store the user's theme preference.
- **language**: String for the preferred language.

THANKYOU