# Project Name: calculator App

## Name :sachin

# Document Version Control

| Date issued | Version | Description | Author |
|---|---|---|---|
| July 15, 2024 | 1.1 | First Draft | sachin |
| July 25, 2024 | 1.2 | Added animations | sachin |
| July 27, 2024 | 1.3 | Correcting font sizes | sachin |

# 1 Abstract

ReactJS is a part based library which is conveyed for the advancement of intelligent UIs. Presently it is the most famous front-end JS library. It consolidates the view (V) layer in M-V-C (Model View Controller) design. It is bolstered by Facebook, Instagram and a network of individual designers and associations. Respond fundamentally empowers advancement of enormous and complex online applications which can change its information without ensuing page revives. It focusesto furnish better client encounters and with blasting quick and powerful web applications advancement. ReactJS can likewise coordinated with other JavaScript libraries or structures in MVC, for example, AngularJS.

A calculator is a device that performs arithmetic operations on numbers. Basic calculators can do only addition, subtraction, multiplication and division mathematical calculations.

5

# 1 Introduction

A calculator is a device that performs arithmetic operations on numbers. Basic calculators can do only addition, subtraction, multiplication and division mathematical calculations.

The most basic calculator is the four-function calculator, which can perform basic arithmetic such as addition, subtraction, multiplication and division.

These are sometimes called pocket calculators or hand-held electronic calculators because they are small enough to fit in a shirt pocket. They are also the least expensive calculator, costing around $5 or less.

Four-function calculators usually have a +, -, x and / sign to denote the operations and can produce decimal numbers. Some also have a % button, which is used to calculate percentages.

## 1.2 scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

## 1.3 Definitions

| Term | Description |
|---|---|
| UGV | Unmanned Ground Vehicle |
| Database | Collection of all the information monitored by this system |
| IDE | Integrated Development Environment |

# 2 General Description

## 2.1 Product Perspective

Unlock the power of precision with the PrecisionPro Calculator, your ultimate companion for academic, professional, and everyday calculations. Designed to cater to a wide range of mathematical needs, this high-performance calculator features an intuitive interface and a robust set of functions, making complex calculations straightforward and efficient.

## 2.2 Problem statement

In today's fast-paced academic and professional environments, users demand calculators that not only perform basic arithmetic but also handle complex mathematical operations with ease. However, many existing calculators fall short in areas such as user interface design, functionality, and adaptability to various user needs. This gap affects productivity and accuracy, leading to frustrations among users.

## 2.3 PROPOSED Solution

To address the challenges identified, we propose the development of an Advanced Multi-Function Calculator that combines cutting-edge technology with user-centric design. This solution aims to enhance usability, expand functionality, improve durability, and optimize power management to meet the diverse needs of users.

## 2.4 FURTHER IMPROVEMENTS

Certainly! Here are additional enhancements to take the Advanced Multi-Function Calculator to the next level, focusing on further improving user experience, functionality, and integration with modern technology:

## 2.5 Technical Requirements

The technical requirements for designing and developing a calculator encompass various aspects including hardware, software, and integration capabilities. Here’s a detailed breakdown of the mechanical and technical requirements for a high-performance calculator:

## 2.6 Data Requirements

When designing and developing a calculator, data requirements encompass various aspects related to the functionality, user interaction, and data management. Here's a comprehensive overview of the data requirements for a calculator:

1. **Functional Data:**

**a. Mathematical Functions:**

- **Basic Arithmetic:** Addition, subtraction, multiplication, and division.
- **Advanced Functions:** Exponential, logarithmic, trigonometric (sine, cosine, tangent), and inverse trigonometric functions.
- **Statistical Functions:** Mean, median, mode, standard deviation, variance, and correlation.
- **Algebraic Functions:** Polynomial equations, roots, and complex numbers.
- **Calculus Functions:** Derivatives, integrals, and limits.
- **Financial Functions:** Interest rates, annuities, and depreciation.

**b. User-Defined Functions:**

- **Custom Formulas:** Ability to create and store user-defined formulas and functions.
- **Variable Storage:** Management of user-defined variables and constants.

**c. Data Input and Output:**

- **Input Formats:** Support for various input formats, including scientific notation and algebraic expressions.
- **Output Formats:** Display of results in different formats, such as decimal, fraction, or percentage.
- **2. User Data:**
- **a. Personalization:**
- **User Profiles:** Storage of individual user settings, preferences, and customizations.
- **Custom Menus and Shortcuts:** Ability to save and retrieve user-defined menus and shortcuts.
- **b. Calculation History:**
- **History Management:** Storage of previous calculations and results for reference and review.
- **Search and Retrieval:** Functionality to search and retrieve past calculations.
- **c. Data Backup and Syncing:**
- **Cloud Sync:** Ability to sync user data, settings, and customizations across multiple devices via cloud storage.

- **Backup Options:** Regular backups of user data to prevent loss in case of device failure.

## 2.7 Tools used

Multiple libraries and tools have been used for this project whether its for UI , animation or some other aspect of the project . They are as given below –

HTML (HyperText Markup Language) is essential for creating the structure of web pages, including calculators. Here's how HTML is used to build a basic web-based calculator and the key HTML elements involved:

CSS (Cascading Style Sheets) is crucial for styling and laying out HTML elements to create an aesthetically pleasing and functional calculator. Here's how CSS can be used to enhance the appearance and usability of a web-based calculator:

GitHub is a powerful platform for version control and collaboration, making it an excellent tool for managing the development of a calculator or any other software project. Here's how GitHub can be utilized effectively throughout the development process of a calculator:

javaScript is essential for adding interactivity and functionality to a web-based calculator. It handles user input, performs calculations, and updates the user interface dynamically. Here's how JavaScript is typically used in the development of a calculator, including key concepts and example code.

Using React for building a calculator can significantly enhance the development process due to React's component-based architecture, state management, and reactivity. Here's a detailed guide on how to build a simple calculator using React.js.

High Level Design

2.8 Constraints

When developing a calculator, there are several constraints and challenges you might face. These constraints can impact both the design and functionality of the calculator, and addressing them effectively is crucial for creating a robust and user-friendly application. Here's a comprehensive look at various constraints:

**User Interface Constraints:**

**a. **Screen Space:**

- **Constraint:** Limited space on the screen can restrict the number of buttons and display elements.
- **Solution:** Use responsive design principles to adjust the layout for different screen sizes. Consider a grid layout to organize buttons efficiently.

. Functionality Constraints: a.

**Basic Operations:**

- **Constraint:** Limited to basic arithmetic operations (addition, subtraction, multiplication, division) unless extended functionality is provided.
- **Solution:** Implement additional functionalities such as square roots, percentages, or scientific calculations if needed.
-

## 2.9 Assumptions



When developing a calculator, several assumptions are typically made to simplify design and implementation. These assumptions help define the scope of the project and guide the development process. Here are some common assumptions:

****1. Basic Functional Assumptions:**

**a. **Arithmetic Operations:**

- **Assumption:** The calculator will handle basic arithmetic operations such as addition, subtraction, multiplication, and division.
- **Impact:** Limits the initial scope to fundamental operations, possibly excluding more advanced mathematical functions.

**b. **Single Expression Calculation:**

- **Assumption:** The calculator processes one expression at a time (e.g., 3 + 5), rather than handling multiple sequential operations without intermediate steps.
- **Impact:** Simplifies the logic by focusing on evaluating complete expressions.

****2. Input Handling Assumptions:**

**a. **Valid Input Format:**

- **Assumption:** Users will input valid expressions and numbers (e.g., 2 + 2, 3.5 * 4), and invalid inputs will be handled or ignored.
- **Impact:** Focuses on typical user behavior and error handling, potentially limiting validation for malformed inputs.

**b. **No Complex Syntax:**

- **Assumption:** The calculator does not need to handle complex mathematical syntax or functions beyond basic arithmetic (e.g., trigonometric functions, logarithms).
- **Impact:** Keeps the implementation straightforward, but may limit functionality.

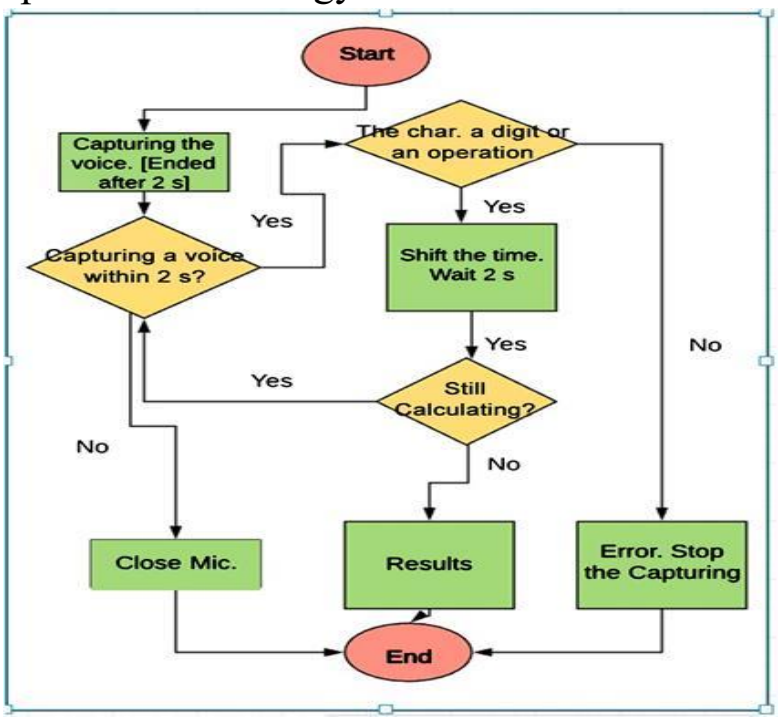
IC

# 3 Design Details

## 3.1 Process Flow

The process flow of a calculator describes the sequence of steps from the user's input to the final output. This flow involves interaction between the user interface, the application's logic, and the display. Below is a detailed description of the typical processed flow of a calculator:

Proposed methodology



## 3.2 Event log

An event log for a calculator typically includes records of various activities and operations performed using the calculator. While calculators usually don't have a built-in feature for logging events in the way computers or software might, a hypothetical event log for a digital calculator could look something like this:

## 3.3 Error Handling

Error handling in calculators involves managing and responding to conditions where operations can't be completed as intended. Here are common errors in calculators and how they are typically handled:

**1. Division by Zero**

- **Error Message**: "Error" or "Cannot divide by zero"
- **Handling**: The calculator stops the operation and displays an error message to indicate that division by zero is undefined.

**2. Overflow**

- **Error Message**: "Overflow" or "Error"
- **Handling**: This occurs when a calculation exceeds the calculator's numerical limits. The calculator displays an error message to indicate that the result is too large to compute.

**3. Invalid Input**

- **Error Message**: "Syntax Error" or "Error"
- **Handling**: This happens when the user inputs an incorrect sequence of operations or symbols. The calculator prompts the user to correct the input.

# 4 Performance

The performance of a calculator can be assessed in various ways depending on the type of calculator and its intended use. Here's a breakdown of key performance aspects for different types of calculators:

**1. Basic Calculators Performance**
**Aspects:**

- **Speed**: Basic calculators are designed for rapid arithmetic operations (addition, subtraction, multiplication, and division). They usually perform these operations almost instantaneously.
- **Accuracy**: High accuracy for standard arithmetic operations is expected. Any deviation might be due to hardware malfunction or software issues. • **Reliability**: Should consistently deliver correct results for simple calculations.

## 4.1 Reusability

The reusability of a calculator, in a broad sense, refers to its ability to be used multiple times for various tasks or functions without losing effectiveness. This concept can be applied differently depending on the type of calculator

**1. Basic Calculators**

**Reusability Aspects:**

- **Functionality**: Can be used repeatedly for basic arithmetic tasks such as addition, subtraction, multiplication, and division.

- **Durability**: Typically designed for long-term use with minimal maintenance. Their durability often contributes to their high reusability.

- **Simplicity**: Easy to use and maintain, which supports consistent reusability without requiring specialized knowledge or frequent updates.

**Typical Use Cases**: Everyday calculations, quick math tasks.

## 4.2 Deployment

**Physical Calculators**

**Deployment Aspects:**

- **Manufacturing and Distribution**: Physical calculators need to be manufactured and then distributed through retail channels or directly to consumers. This includes packaging, labeling, and logistics.
- **Retail and Sales Channels**: Establish partnerships with retailers, online marketplaces, or educational suppliers to make the calculators available for purchase.
- **In-Store Display**: For retail environments, ensure that calculators are displayed effectively to attract buyers, often requiring point-of-sale (POS) placement and marketing materials.

- **Training and Support**: Provide training materials or guides for users to understand the calculator's features and functionalities. Support may be needed for troubleshooting and maintenance.

# 5 Dashboards

**Basic Calculators**
For basic calculators, dashboards are typically minimal or non-existent, as these calculators focus on simple arithmetic tasks. However, if a basic calculator app includes a dashboard, it might include:

- **History Panel**: Displays recent calculations or a log of past operations.
- **Quick Access Buttons**: Shortcuts for frequently used functions or settings. • **Settings Menu**: Options to customize the calculator's appearance or behavior.

## 5.1 KPIs (Key Performance Indicators)

1 . Key indicators displaying a summary of the anomaly detection in the society/area.

2. Time and workload reduction using the UGV based surveillance.

3. To detect mob (illegal) activities and inform police.

4. On time alert to nearest hospital on medical emergency (accident).

5. Taking adequate evidence of mob.

6. Send disaster details to concerned authorities.

7. Display of battery life and percentage of UGV.

8. Distance travelled by UGV.

9. Get the exact location of UGV.

# 6 Conclusion

In conclusion, calculators, whether physical devices or digital applications, are essential tools for performing arithmetic, scientific, financial, and complex mathematical operations. Their design, functionality, and usability significantly impact their effectiveness and applicability in various contexts.

**Basic Calculators**: Ideal for simple arithmetic tasks. They are durable, easy to use, and costeffective, making them suitable for everyday calculations.