Calculator App :Using ReactJs

# Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date Issued | Version | Description | Author |
| Date | 1 | Initial HLD — V1.0 | Name |
| Date | 2 | Updated KPI — V1.1 | Name |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Contents

[Document Version Control 2](#_bookmark0)

1. [Introduction 4](#_bookmark2)
   1. [Why this High-Level Design Document? 4](#_bookmark3)
   2. [Scope 4](#_bookmark4)
   3. [Definitions 4](#_bookmark5)
2. [General Description 5](#_bookmark6)
   1. [Product Perspective 5](#_bookmark7)
   2. [Problem statement 5](#_bookmark8)
   3. [PROPOSED SOLUTION 5](#_bookmark9)
   4. [FURTHER IMPROVEMENTS 5](#_bookmark10)
   5. [Technical Requirements. 5](#_bookmark11)
   6. [Data Requirements 7](#_bookmark13)
   7. [Tools Used 8](#_bookmark14)

2.7.1 Hardware Requirements…………………………………………………………..9

* 1. [Constraints. 10](#_bookmark15)
  2. Assumptions…………………………………………………………………………….10

1. [Design Details 11](#_bookmark16)
   1. [Process Flow 11](#_bookmark17)
   2. [Error Handling 12](#_bookmark21)
2. Performance 13
   1. Reusability 13
   2. Application Compatibility……………………………………………………………….13
   3. Resource Utilization 13
3. Conclusion 14
4. **Introduction**

##### Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

* + - Present all of the design aspects and define them in detail
    - Describe the user interface being implemented
    - Describe the hardware and software interfaces
    - Describe the performance requirements
    - Include design features and the architecture of the project
    - List and describe the non-functional attributes like: o Security
      * Reliability
      * Maintainability
      * Portability
      * Reusability
      * Application compatibility
      * Resource utilization
      * Serviceability

#### 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. **Definitions**



*Term UGV*

*Database*

*IDE AWS*

*Description*

Unmanned Ground Vehicle

Collection of all the information monitored by this system

Integrated Development Environment

Amazon Web Services

1. General Description

### Product Perspective

### The ReactJS Calculator App is designed to provide users with a user-friendly tool for performing basic arithmetic calculations quickly and conveniently. It serves as a digital replacement for traditional physical calculators, offering a simple and efficient solution for users who need to perform calculations on their computers or mobile devices.

* 1. **Problem statement**
* Build a simple calculator app which performs simple arithmetic operations such as addition, subtraction, multiplication and division
* Create a visually appealing and intuitive user interface using React components, HTML, and CSS.
* Display buttons for numbers (0-9) and common arithmetic operations (addition, subtraction, multiplication, division).
* Allow users to input numerical values by clicking on number buttons.

###### PROPOSED SOLUTION

The ReactJS Calculator App will be developed to provide users with a user-friendly and modern calculator interface for performing basic arithmetic calculations.

###### FURTHER IMPROVEMENTS

Implement a scientific mode that offers advanced mathematical functions such as trigonometric, logarithmic, and exponential calculations. This can extend the app's utility for users who require more advanced calculations.

Enable keyboard input for the calculator app. Users can input numbers, operators, and perform calculations using keyboard shortcuts, enhancing accessibility and usability.

### Technical Requirements

**Environment Setup:**

* Install Node.js and npm (Node Package Manager) on your development machine.
* Set up a code editor (e.g., Visual Studio Code) for writing and editing your React code.

**ReactJS Project Setup:**

* Initialize a new React project using create-react-app or another React boilerplate.
* Organize your project structure with components, assets, styles, and tests.

**User Interface:**

* Create a responsive and user-friendly UI using HTML and CSS within React components.
* Design buttons for numbers (0-9), arithmetic operators (+, -, \*, /), and special functions.
* Implement a display area to show input values, selected operations, and calculated results.

### Data Requirements

Input Values:

* Numerical Values: The app needs to capture numerical values entered by the user through button clicks or keyboard input.
* Arithmetic Operators: The app needs to capture the selected arithmetic operator (addition, subtraction, multiplication, division).

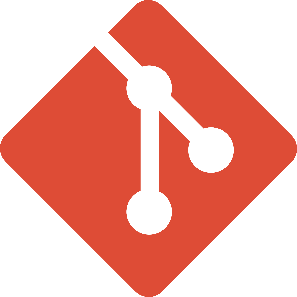
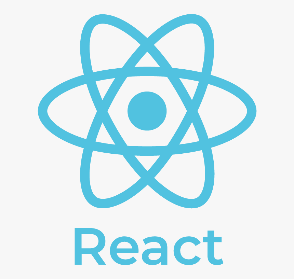
State Management:

* Current Input: The app needs to maintain the current input values (e.g., digits and operators) as the user interacts with the calculator.
* Selected Operator: The app needs to store the currently selected arithmetic operator.
* Calculated Result: The app needs to store the result of the calculation for display purposes.

History and Memory :

* Calculation History: If you implement a history panel, you'll need to store a log of recent calculations.
* Memory Values: For memory functions, the app needs to store values in memory registers.
  1. Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, TensorFlow, Keras and Roboflow are used to build the whole model.



* + - VSCode is used as IDE.
    - Front end development is done using HTML/CSS
    - HTML is used as structure of a website
    - CSS is used for designing the website
    - Reactjs is used for proper functioning of a calculator .
    - GitHub is used as version control system.
    1. Hardware Requirements

Processor (CPU):

• A modern dual-core or quad-core processor is sufficient for running a ReactJS Calculator App smoothly.

Memory (RAM):

• At least 4GB of RAM is recommended for optimal performance. More RAM can be beneficial if you have other applications running simultaneously.

Storage:

• The storage requirements for a ReactJS Calculator App are very minimal, as it mainly involves storing code files and assets. A few hundred megabytes of storage space are more than enough.

Web Browser:

• You'll need a modern web browser (e.g., Chrome, Firefox, Safari, Edge) to access and test your ReactJS Calculator App.



### Constraints

The app's performance might be impacted by factors such as complex calculations or suboptimal code structure.

### Assumptions

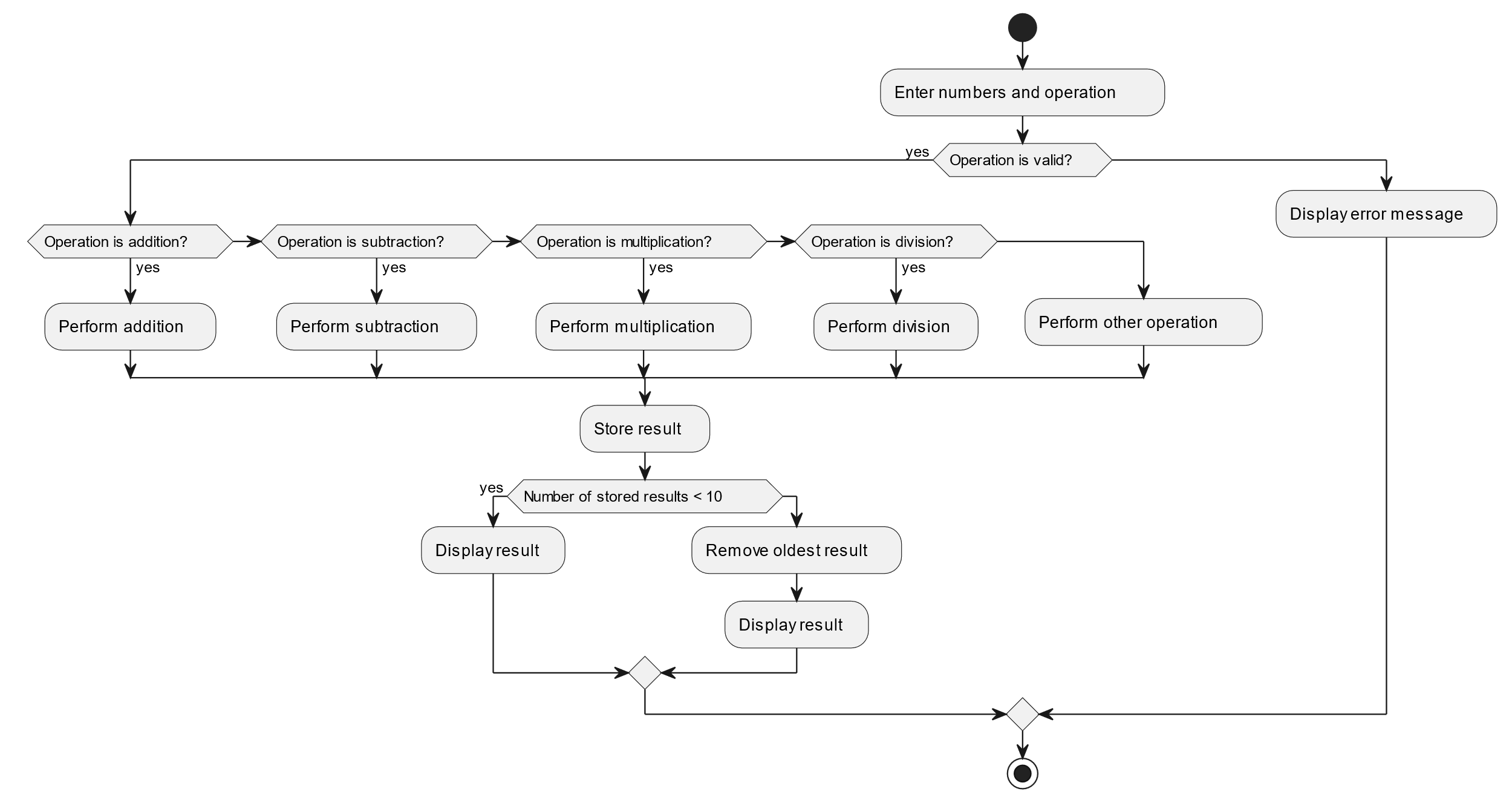
* Basic Arithmetic Operations: You assume that the calculator app will primarily focus on supporting basic arithmetic operations such as addition, subtraction, multiplication, and division.

## Design Details

#### Process Flow

For identifying the different types of anomalies, we will use a deep learning base model. Below is the process flow diagram is as shown below.

#### Proposed methodology



* 1. **Error Handling**
* Validate user input to ensure that only valid numerical values and operators are accepted. Prevent users from entering characters that are not part of the allowed input set.
* Implement logic to handle division by zero. When the user attempts to divide by zero, display an error message and prevent the calculation.
* Handle cases where the user enters consecutive operators (e.g., "5 ++ 3") or starts with an operator (e.g., "\* 5"). Display an error message to guide the user.

1. Performance

• Measure the time it takes for the app to load and become usable after the user opens it in a web browser. Faster loading times lead to better user experience.

* Evaluate how quickly the app responds to user interactions, such as clicking buttons or entering input. A responsive app provides immediate feedback to users.

### Reusability

Creating a ReactJS Calculator App with a focus on reusability is a commendable approach. By designing the app as a reusable component, you ensure that its core functionality can be easily integrated into various projects without reinventing the wheel.

### Application Compatibility

Ensuring application compatibility is crucial for a reusable component like the ReactJS Calculator App. By adhering to best practices in coding standards and using modern web technologies, you increase the chances of compatibility across a wide range of web browsers and devices

### Resource Utilization

Efficient resource utilization is pivotal for delivering a performant user experience. The ReactJS Calculator App should be designed to be lightweight and optimized, taking up minimal memory and processing power.

## Conclusion

The High-Level Design for the Calculator App outlines the app's architecture, features, user interface, and functionality flow. This design serves as a blueprint for the subsequent Low-Level Design and implementation phases.

Note: This is a simplified example. Your actual HLD should be more detailed and specific, outlining the interactions between different components, dependencies, and any design patterns you plan to use