Low Level Design (LLD)

React Js : Calculator

Abhishek Shinde

**Contents**

[**Abstract** 3](#_bookmark1)

[1](#_bookmark2) Introduction 3

[1.1](#_bookmark3) Why this LLD document 3

[1.2](#_bookmark4) Scope 4

[1.3](#_bookmark5) Constraints 4

[1.4](#_bookmark6) Risks 4

[1.5](#_bookmark7) Out of Scope 4

[2](#_bookmark8) Technical Specifications 5

* 1. [Components 5](#_bookmark9)

2.1.1 Input Schema 6

* 1. [Deployment 7](#_bookmark10)

[3](#_bookmark12) Technology Stack 7

[4](#_bookmark13) Proposed Solution 7

[5](#_bookmark14) Workflow 8

[6](#_bookmark15) User I/O workflow 9

[7](#_bookmark16) Test cases 10

[7](#_bookmark16) KeyPerformanceIndicator(KPI) 10

# Abstract

The ReactJS Calculator Web App is a modern and user-friendly application designed to empower users with the ability to perform basic arithmetic calculations through an intuitive and responsive interface. Leveraging the power of the React JavaScript library, this web app offers a versatile and seamless experience for individuals seeking a reliable tool for everyday calculations.

# Introduction

## Why this Low-Level Design Document?

The purpose of this Low-Level Design (LLD) document is to provide an in-depth technical blueprint for the implementation of the ReactJS Calculator App. While the High-Level Design offers an overview of the app's architecture and components, this LLD document delves deeper into the specifics of how the app will be structured, the interactions between components, data flow, and error handling mechanisms.  
  
By detailing the finer technical aspects of the app, this document serves as a comprehensive reference for developers who will be involved in the implementation phase. It outlines the decision-making process behind the chosen design patterns, state management approach, and user interface interactions. Moreover, the LLD document acts as a guiding resource for ensuring code modularity, maintainability, and reusability throughout the development lifecycle.

This project shall be delivered in two phases:

Phase 1: Creating all basic Arithmetic Operations

Phase2: Creating both Light and Dark Mode in Calculator App.

## Scope

The scope of the ReactJS Calculator App encompasses the development of a user-friendly web-based calculator application that provides users with the ability to perform basic arithmetic calculations. The app will be designed to offer a straightforward and intuitive interface for users to input numerical values and select arithmetic operators for addition, subtraction, multiplication, and division operations. The primary goal of the app is to facilitate quick and accurate calculations while maintaining a responsive and visually appealing user experience.

## Constraints

Limited project timeline might restrict the extent to which you can implement advanced features or thoroughly test the app.

## Risks

Risk: The app might render differently or have functionality issues on various web browsers, impacting user experience.

## Out of Scope

Scientific Calculations . High Level Mathematical Computations.

# Technical specifications

## Components

## 1. HTML Structure

## - Defines buttons for numbers, operators, and equal sign.

## - Defines a display area to show input and output.

## 2. CSS Styling

## - Provides styling for buttons, input display, and overall layout.

## 3. JavaScript Logic

## Functions:

## - `init()`: Initializes event listeners for button clicks.

## - `handleNumberClick(number)`: Captures the clicked number and updates the display.

## - `handleOperatorClick(operator)`: Captures the clicked operator and performs the calculation.

## - `calculateResult()`: Performs the calculation based on the input and operator.

## - `updateDisplay(content)`: Updates the display area with the provided content.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator name** | **Datatype** | **Operation** | **Result** |
| Addition | int | 3+2 | 5 |
| Addition | float | 7.20+0.1 | 7.30 |
|  |  |  |  |

* + 1. **Input schema**

## Deployment

1. Github Pages



# Technology stack

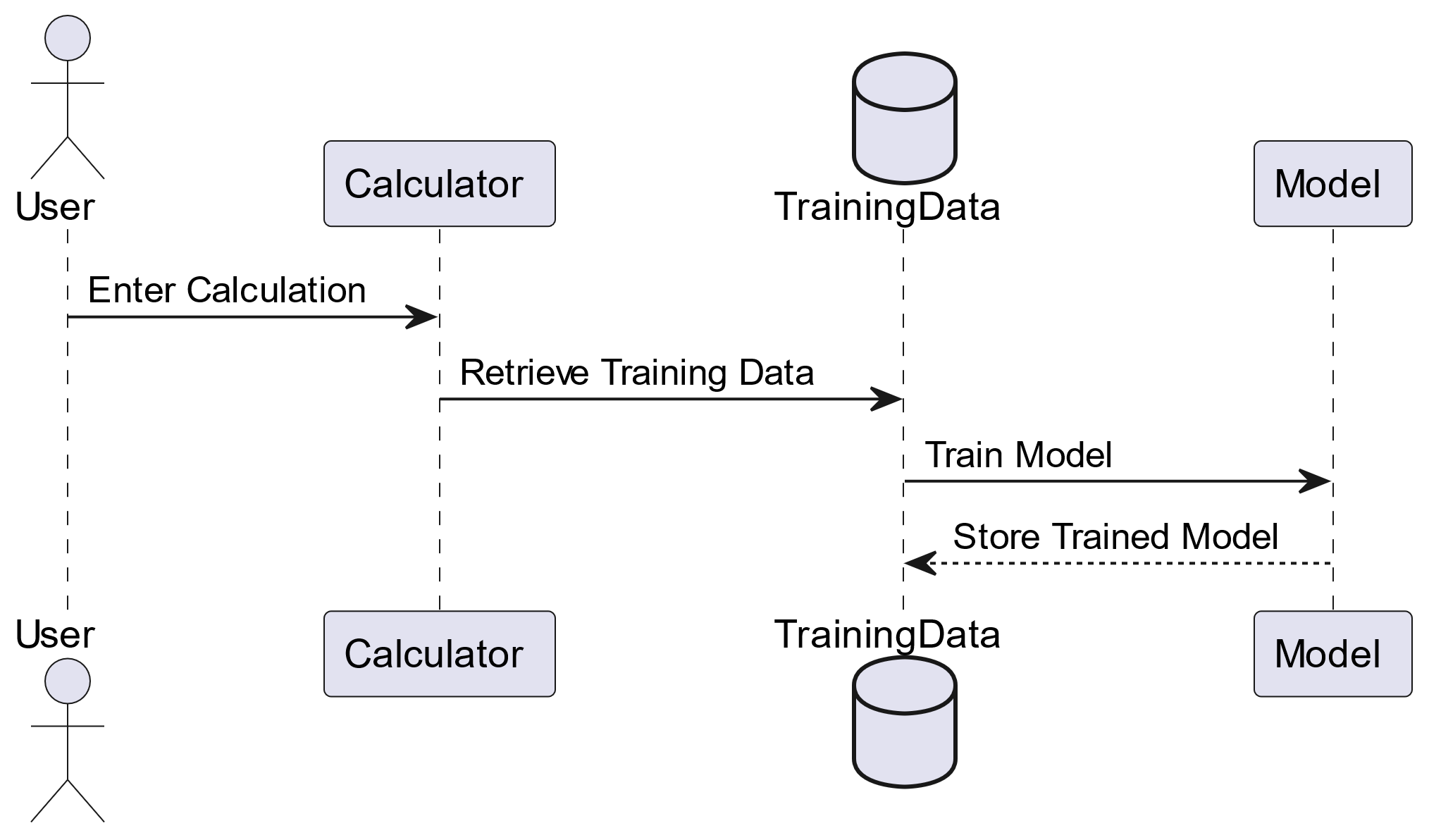
|  |  |
| --- | --- |
| **Front End** | HTML/CSS/JS/React |
| **Backend** | No Backend |
| **Database** | No Database |
| **Deployment** | Github Pages |

1. **Proposed Solution**

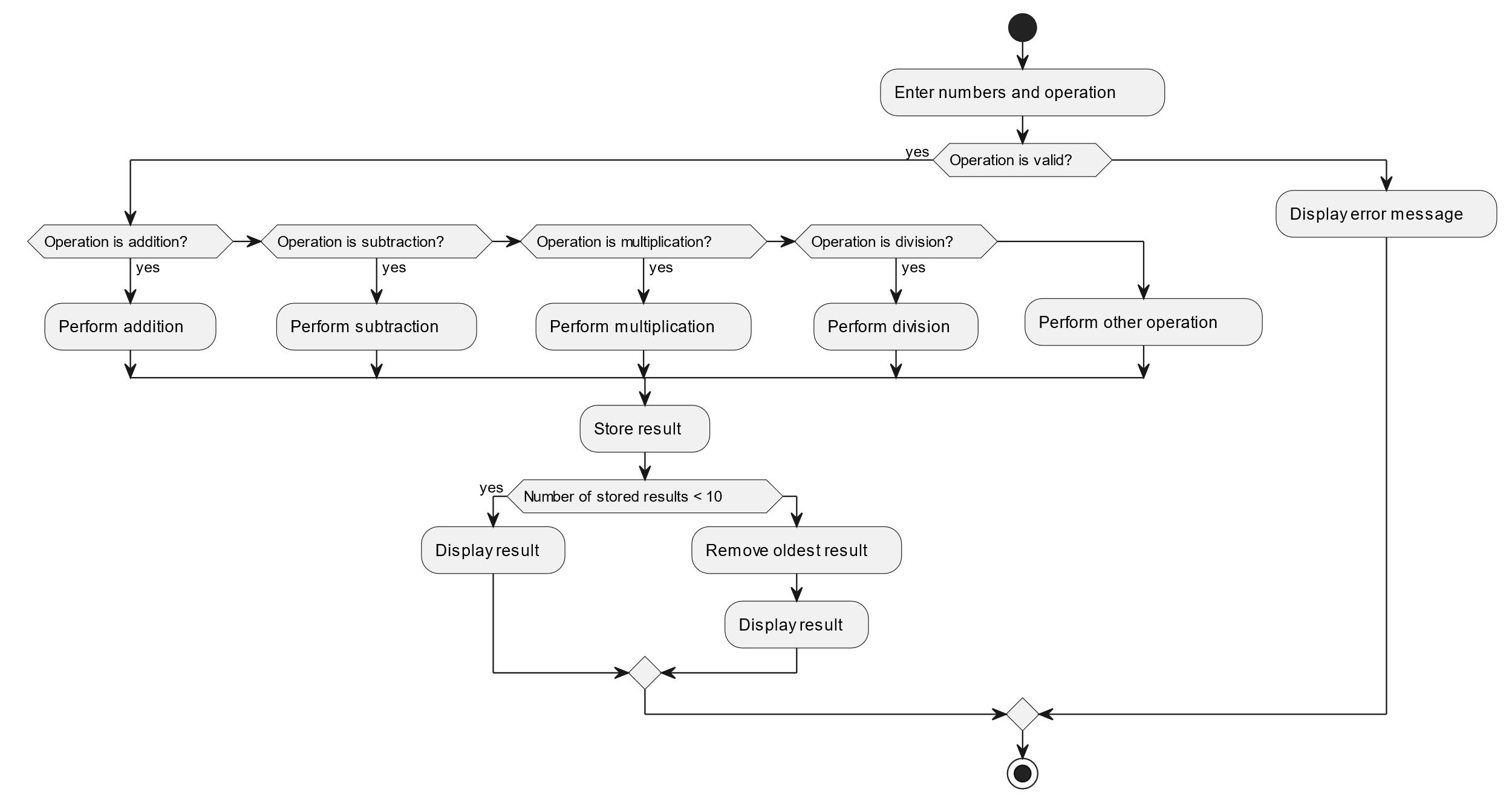
The proposed solution for the ReactJS Calculator Web App involves the development of a modular and user-friendly calculator application that enables users to perform basic arithmetic operations. The application will be built using the React JavaScript library to provide a responsive and interactive user interface. The solution will adhere to the principles of modularity, maintainability, and reusability, allowing the calculator component to be integrated seamlessly into various web projects.

By following this proposed solution, we aim to create a high-quality ReactJS Calculator Web App that fulfills its intended purpose while adhering to best practices in design, development, and user experience.

# Model training/validation workflow



1. **User I/O workflow**

****

1. **Test cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case** | **Steps to perform test case** | **Module** | **Pass/Fail** |
| 1 | Divide any number by  ‘0’. | Successfully it shows the message ‘Infinity’ | Pass |

# Key performance indicators (KPI)

* Measure the time it takes for the app to load and become fully interactive. Lower loading times contribute to a better user experience.
* Track the time it takes for the user to be able to interact with the app after it starts loading. Faster TTFI indicates quicker usability.
* Track the time it takes for the user to be able to interact with the app after it starts loading. Faster TTFI indicates quicker usability.
* Track the number of times the app's pages are viewed. This metric gives an indication of the app's popularity and usage.