Calculator Project

Software Requirements Specifications

Version <1.0>

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 10/11/23 | 1.0 | Started Software Requirements Specifications | Barret Brown, Ethan Doughty, Minh Vu, Adam Berry, Bisshoy Bhattacharjee |
| 10/15/23 | 1.0, | Completed Software Requirements Specifications | Ethan Doughty, Minh Vu |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Overall Description 5

2.1 Product perspective 5

2.1.1 System Interfaces 5

2.1.2 User Interfaces 5

2.1.3 Hardware Interfaces 5

2.1.4 Software Interfaces 5

2.1.5 Communication Interfaces 5

2.1.6 Memory Constraints 5

2.1.7 Operations 5

2.2 Product functions 5

2.3 User characteristics 5

2.4 Constraints 5

2.5 Assumptions and dependencies 5

2.6 Requirements subsets 5

3. Specific Requirements 5

3.1 Functionality 5

3.1.1 <Functional Requirement One> 6

3.2 Use-Case Specifications 6

3.3 Supplementary Requirements 6

4. Classification of Functional Requirements 6

5. Appendices [6](#_qsh70q)

Software Requirements Specifications

# Introduction

## Purpose

The purpose of this document is to describe the external behavior of the application, as well as describe the nonfunctional requirements, design constraints, and other specifications that are needed for the software to perform correctly.

## Scope

The SRS applies to the calculator that this group will be building. The calculator should be able to perform

basic PEMDAS operations based on user inputs.

## Definitions, Acronyms, and Abbreviations

RAM - Random Access Memory

PEMDAS - Parentheses, Exponent, Multiplication, Division, Addition, Subtraction

CLI - Command Line Interface

GitHub- A code hosting platform for version control and collaboration

UI - User Interface

SRS - Software Requirements Specification

## References

## Overview

The rest of this document contains the overall description, specific requirements and the functional requirements for the software

# Overall Description

## Product Perspective

### System Interfaces

* It is self-contained so it will have no need to connect to outside systems.

### User Interfaces

* Numeric buttons
* Operator buttons
* History/Clear history

### Hardware Interfaces

* Since the program is run on the computer there will be no need to interface with any external hardware

### Software Interfaces

* The project will be self-contained and will not interface with any outside software.

### Communication Interfaces

* The project will not have any communication interfaces as it will all be local on the machine and have no need to communicate outside of itself.

### Memory Constraints

* The project will be low in size so it will take up a minimal amount of disk space and also use a minimal amount of RAM.

### Operations

## Product functions

## User characteristics

* Software is intended for the general public with varying degrees of technical knowledge and backgrounds so the program should be more intuitive and not require an upper level of technical knowledge to setup/use

## Constraints

* Programmed in C++
* Stored/Collaborated on through Github

## Assumptions and dependencies

* There will be 5 members on the team developing the arithmetic expression evaluator
* This will be a semester-long project, with each release of this project expected to be completed a couple of days before the due date to leave enough time for quality control and review

## Requirements subsets

* Functional Requirements
  + Essential (Required for project to work)
  + Desirable (Not required for project but wanted to improve project)
  + Optional (Not required and not needed to improve project but helps it)
* Non-functional Requirements
* Constraints

# Specific Requirements

## Functionality

***Expression Parsing:***

* This involves creating a stack data structure paired with a function to token the expression given to split up the expression into multiple smaller to handle expressions

***Operator Precedence:***

* Follows the order of PEMDAS when solving equations

***Arithmetic Computation:***

* The process itself of solving an equation i.e: the function adds 5+5 and returns 10

***Numeric Constants:***

* Implementing logic to handle numeric constants, i.e: 5(10\*2) returns 100 not 520 or 5(20)

***Parentheses Handling:***

* Handles excess parentheses and parses through them to “get” the equation incased i.e: would handle problems like (((((5\*2+(10/2))))))

***Error Handling:***

* Handle any divide by 0 equations or other equations incorrectly inputted i.e: j+5 or 10/0

***User Interface:***

* Some type of user interface to allow for easy input for users, either through a CLI or another system.
  + If not CLI, have a similar layout to a “normal” calculator so as to help approachability

***Evaluation of floating point numbers:***

* Allow handling outside of just integers i.e.: 1.5\*10 returns 15

***Handling of white space:***

* If the equation has white space in it it will be removed prior to any parsing not have errors. i.e.: (5 + 10) would turn into (5+10)

***Memory Recall:***

* Allow for user to recall the history of equations/answers entered before i.e.: 5(20+5)=125
  + Maybe allow for a “clearing” of history either by a command/button or only store with launch

***Alternative Symbols for common expressions:***

* Allowing for power symbol to be \*\* and brackets to be used in place of parentheses.

## Use-Case Specifications

* Use case specifications will have a description, precondition, multiplicities, and a flow pattern

## Supplementary Requirements

* Keep the total runtime low by making sure code is written for compile time instead of run time to reduce overhead on user calculations.
* Reliable, if the user enters an equation that has errors make sure it doesn’t crash the program or create lag.
* The User interface should be easy to use and intuitive
* The program should be “portable” and easy to set up on a machine (preferably 1 click on a file for user)
* Coded in C++
* Code versions stored/collaborated on through GitHub

# Classification of Functional Requirements

| **Functionality** | **Type** |
| --- | --- |
| Expression Parsing | Essential |
| Operator Precedence | Essential |
| Arithmetic Computation | Essential |
| Numeric Constants | Essential |
| Parentheses Handling | Essential |
| Error Handling | Essential |
| User Interface | Desirable |
| Evaluation of floating point numbers | Desirable |
| Handling of white space in equation | Desirable |
| Brackets have the same behavior as parentheses | Optional |
| Alternative Powers | Optional |
| Memory recall | Optional |

# Appendices