<Company Name>

QuickMath.io

Software Requirements Specifications

Version 1.0

Revision History

Date	Version	Description	Author
27/09/23	<0.1>	Split tasks and added information	Vinny, Omar, Tatum, David, Owen, Jamie
1/10/23	<0.2>	Added further information	Omar
10/11	<0.3>	Add 2.1 section information	Vinny
10/12	<0.4>	Add section 3, 4, and 5 information	Owen, Tatum
10/12	1.0	Proofread and finalized	David, Omar, Vinny, Tatum, Owen, Jamie

Table of Contents

Software Requirements Specifications

1. Introduction

This Software Requirements Specification (SRS) comprehensively documents all software requirements for the arithmetic expression evaluator system to be developed using use-case driven methodology. This SRS contains use cases and supplementary specifications that capture the complete functional and non-functional requirements following guidelines from IEEE 830-1998. Section 1 introduces the SRS. Section 2 provides the system overview. Section 3 contains the detailed requirements. Section 4 details functional requirements associated with our system. Appendices provide supporting information. The requirements in this SRS serve as the foundation for the design, implementation, and testing of the system.

1.1 Purpose

The purpose of this SRS document is to unambiguously specify all requirements for an arithmetic expression evaluator program to be developed in C++. The SRS will provide meticulously detailed functional and non-functional requirements that the expression evaluator program must satisfy. It will serve as a comprehensive specification to enable the development team to accurately design, implement, validate, and verify the system to meet all specified requirements.

1.2 Scope

This SRS contains the complete set of requirements exclusively for the arithmetic expression evaluator program to be built as a term project for EECS 348 Software Engineering at the University of Kansas under Professor Hossein Saiedian during the Fall 2023 semester. It encompasses every capability, feature, constraint, and design guide outlined in the project description provided by the professor. The scope applies solely to this term project.

1.3 Definitions, Acronyms, and Abbreviations

SRS - Software Requirements Specification

PEMDAS - Mathematical order of operations (Parentheses, Exponents, Multiplication/Division, Addition/Subtraction)

OOP – Object Oriented Programming

1.4 References

1. EECS 348 Project Description, Version 1.0, Professor Hossein Saiedian, Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, Fall Semester 2023.

1.5 Overview

This SRS document contains all software requirements for the arithmetic expression evaluator program. Section 1 provides the introduction. Section 2 gives a comprehensive technical overview of the product perspective, capabilities, interfaces, and operating context. Section 3 captures all detailed software system requirements categorizing functional and non-functional needs. Section 4 classifies requirements by priority. Appendices are included for supplementary specifications as needed. The requirements catalogued in this SRS establish the foundation for the design, implementation, verification, and validation of the expression evaluator program.

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

The software product will carry out arithmetic calculations based on user input. On a high level, the system is split into two components. The first components which tokenizes the input, and then the second component to carry out the computation.

2.1.2 User Interfaces

There are no exceptional user interfaces, other than a Linux shell to run the executable.

2.1.3 Hardware Interfaces

There are no exceptional hardware interfaces. However, a keyboard and display are required to interact with the software.

2.1.4 Software Interfaces

The software requires a GCC compiler to compile the code. An official Docker Image will be provided to improve portability. Thus, Docker must be installed on the host system.

2.1.5 Communication Interfaces

The software requires a one-time communication interface to pull the latest Docker Image of the product. Alternatively, the code can be compiled at the host server without a communication interface.

2.1.6 Memory Constraints

The software has no memory constraints, but the operating system may impose additional memory constraints during runtime.

2.1.7 Operations

The software can be used in two ways. The first is to download the source code, and manually compile the code. The second method is to pull the latest Docker Image, after configuring Docker on host system, and run the software through Docker.

2.2 Product functions

The software product will have the following functionalities:

1. Evaluating addition
2. Evaluating subtraction
3. Evaluating multiplication
4. Evaluating division
5. Evaluating modulus
6. Evaluating exponentiation
7. Interpreting parentheses to group operations and operands
8. Items 1-7 in the same call
9. Return value to user

2.3 User characteristics

The software product team characterizes users as the following:

- TAs and Professors who will be grading our work
- Students in need of a strong calculator
- These people may or may not be familiar with the inner workings of the software

2.4 Constraints

The software product will have the following constraints:

1. It will be written in C++
2. It will be due by the end of the semester (No date declared yet, but only a rough assumption can be made that this date will be around Dec. 1)
3. The user interface will be done through a command prompt

2.5 Assumptions and dependencies

At this point during its development cycle, the software product will have the following dependencies:

- It will need to utilize the <iostream> header
- It will need to utilize the <cmath> header

For now, we will assume that these are the only necessary headers.

2.6 Requirements subsets

None

3. Specific Requirements

3.1 Functionality

3.1.1 Allow user to input a string

- Using the command line interface, allow the user to input a mathematical expression to evaluate.

3.1.2 Parse the string, using a stack to properly preserve the desired order of operations.

- Convert the expression from infix form to postfix form.

3.1.3 Evaluate the postfix expression, popping from the stack.

- This expression can consist of parenthesis, spaces, numbers, and operators like *, /, -, + for multiplication, division, subtraction, and addition respectively.
  - This expression can consist of parenthesis, spaces, numbers, and operators like *, /, -, + for multiplication, division, subtraction, and addition respectively.

3.1.4 Print the result of the expression to the user.

- Print the original infix expression input by the user back to the user, and then print the result of the expression in an easily readable form.

3.2 Use-Case Specifications

- The user runs the program from a command line interface, and they are prompted to enter an expression to evaluate. The user inputs an expression, which can consist of parenthesis, spaces, numbers, and operators like *, /, -, and +.
- Once the user has finished their expression, they submit it by pressing enter. Then the program will print the result of their expression.

3.3 Supplementary Requirements

- A clean and easily understood output for the user.

4. Classification of Functional Requirements

Functionality	Type
Take in user input for function	Essential
Evaluating addition	Essential
Evaluating subtraction	Essential
Evaluating multiplication	Essential
Evaluating division	Essential
Evaluating modulus	Essential
Evaluating exponentiation	Essential
Evaluating "**" as exponentiation	Desirable
Interpret parentheses within function for group operations and operands	Essential
Allow the user to use brackets and curly braces ([,], {, }) instead of parentheses	Desirable
Interpret all above operations in conjunction with any parentheses	Essential
Expression parsing	Essential
Produce function to tokenize the input expression	Essential
Calculate various numeric constants	Essential
Have a user-friendly command-line interface for the user inputting the function	Essential
Handle various errors for different scenarios within the functions	Essential
Print the output in a clean and organized way	Desirable

5. Appendices

Not Applicable