Git Goblins

Calc-U-Later

Software Requirements Specification

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 10/14/2023 | 1.0 | Created the Software Requirements Specification | Sabeen, James, Brett, Daniel, Kaden, Anna |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

Software Requirements Specifications

# Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the SRS. The SRS captures the complete software requirements for the system, or a portion of the system. The following is a typical SRS outline for a project using use-case modeling. This artifact consists of a package containing use cases of the use-case model and applicable Supplementary Specifications and other supporting information. This document lists the software requirements for the Calc-U-Later a multifunction calculator.

## Purpose

The purpose of the Calc-U-Later is to perform parse and evaluate arithmetic expressions containing operators +, -, \*, /, %, and ^ as well as numeric constants according to the order of operations. The program should be able to handle expressions with parentheses to define precedence and grouping. The program is built using C++ and must be handle any of the following: expression parsing, operator precedence, parathesis, constants, and error handling. It also must be a user friendly and legible interface that allows users to enter expressions.

## Scope

This software implements a calculator by parsing user-input. This includes features such as addition, subtraction, modulo, exponentiation. Additionally, we have to interact with users and any outside organizations who need to access this document to understand documentation.

## Definitions, Acronyms, and Abbreviations

**Expression Parsing:** the ability to parse arithmetic expressions, taking into account operator precedence and parentheses

**Operators:**

* **+** (addition)
* **-** (subtraction)
* **\*** (multiplication)
* **/** (division)
* **%** (modulo)
* **^** (exponentiation)

**Parenthesis Handling:** the ability to identify and handle expressions enclosed within parenthesis to determine the order of evaluation

**Operator Precedence:** the precedence of the operators according to the PEMDAS rules

**Invalid Expressions:**

* **Unmatched Parentheses:** an expression that has unmatched opening and closing parentheses
* **Division by Zero:** an expression that includes dividing by zero
* **Missing Operand:** a missing operand before and/or after an operator

## References

No outside sources were referred to in this document

## Overview

The rest of the document describes a number of functions of our software including product perspective, functionalities, use-case requirements, classification of functional requirements. Additionally, the rest of the document details supplementary and functional requirements of the Calc-U-Later. It includes a detailed list of the functions and their requirements for the project. It follows the order of Product perspective and sub interfaces, specific requirements, functionality, use-case requirements, classification of requirements, and finally appendices.

# Overall Description

This section of the SRS describes the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 3, and makes them easier to understand. Include such items as:

## Product perspective

### System Interfaces

### User Interfaces

### Hardware Interfaces

### Software Interfaces

### Communication Interfaces

### Memory Constraints

### Operations

## Product functions

## User characteristics

## Constraints

## Assumptions and dependencies

## Requirements subsets

# Specific Requirements

This section of the SRS contains all software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. This Section includes the specific use cases that would be listed in a UML document. This includes expression parsing, operator support, parenthesis handling, numeric constants, user interface, error handling, and project guidelines.

## Functionality

This section describes the functional requirements of the system for those requirements that are expressed in the natural language style. This section is typically organized by features, but alternative organization methods may also be appropriate; for example, organization by user or organization by subsystem.

### Expression Parsing

The application will parse arithmetic expressions entered by the user, taking into account operator precedence and parentheses using data structures to represent the expression’s structure.

### Operator Support

The application must be able to perform the following operators: addition, subtraction, multiplication, division, modulo, exponentiation.

### Operator Precedence

The application must be able to define the precedence of the operators according to the PEMDAS rules and implement the logic to evaluate the expression while considering operator precedence.

### Parenthesis Handling

The application must be able to identify and evaluate expressions within parenthesis and determine order of evaluation.

### Numeric Constants

The application must be able to identify constants in the given expressions and correctly calculate the expression.

### User Interface

The application must have a friendly and legible user interface that allows the user to enter expressions that will be correctly calculated.

### Error Handling

The application must implement robust error handling to manage scenarios that are invalid expressions, such as imbalanced parathesis and division by zero.

## Use-Case Specifications

In use-case modeling, the use cases often define the majority of the functional requirements of the system, along with some non-functional requirements.

Functional Requirements: Addition, Multiplication, Subtraction, Division, Order of Operation, Modulo, Exponentation, Processing Numeric Constants, Error handling, Modulo

Non-Functional Requirements: Must complete within a reasonable amount of time, must handle errors, and must correctly evaluate expression.

## Supplementary Requirements

This application must be built in C++ using any environment. The application must also evaluate the expressions correctly and within a reasonable amount of time. It also must be able to handle errors.

# Classification of Functional Requirements

|  |  |
| --- | --- |
| **Functionality** | **Type** |
| Addition | Essential |
| Multiplication | Essential |
| Subtraction | Essential |
| Division | Essential |
| Order of operations | Essential |
| Modulo | Essential |
| Exponentiation | Essential |
| Processing Numeric Constants | Essential |
| Error handling | Essential |

# Appendices

N/A