Arithmetic Expression Evaluator in C++

Software Requirements Specifications

Version 1.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 10/09/2023 | 1.0 | The team members have provided comprehensive explanations and detailed descriptions for Part 2, which pertains to the software requirements. | Gregory Markose, Taha Khalid, Siddh Bharucha, Saurav Renju, |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 aim of this project is for <company name> to create a versatile arithmetic expression evaluator using

C++. The primary goal is to develop a program that simulates a calculator capable of processing input expressions and performing mathematical operations following the order of operations (PEMDAS). Moreover, <company name> intends to seamlessly incorporate the Software Development Process by delivering various project milestones within the semester. These milestones encompass a comprehensive project plan, a detailed requirements document, a design document that aligns with the specified requirements, a suite of test cases, and a fully functional end product. Within the project scope, it is essential that each phase adheres to a predefined timeline while ensuring alignment with the specified functional requirements.

## Scope

This Software Development Plan is a key component of the larger Arithmetic Parser project. It

encompasses project management processes, objectives, development strategies, role assignments, iterative phases, and provides in-depth descriptions of each iteration, all of which are outlined in the Iteration Plan.

The plans outlined in this document are derived from the project's specific requirements, as detailed in the Requirements Document accessible on Canvas.

## Definitions, Acronyms, and Abbreviations

Can view the Project Glossary

## References

The Software Requirements Specification refers to the following artifacts:

1. Project Plan

2. Iteration Plans

3. Project Management Plan

4. Requirements Document

5. Design Document

6. Test Cases

7. Glossary

## Overview

The Software Requirements Specifications document includes the following content:

* 1. Overall Description: This section offers a summary of the project's purpose, extent, and goals. It also delineates the expected project deliverables. Additionally, this document serves as a comprehensive reference for presenting the precise requirements crucial for the Arithmetic Parser.
  2. Specific Requirements: The SRS defines both the functional and non-functional requirements indispensable for the project's successful fulfillment.
  3. Categorization of Functional Requirements: This part elucidates the various categories of requirements incorporated within the Arithmetic Parser, encompassing essential, preferred, and discretionary requirements.

# Overall Description

## Product perspective

### **System Interfaces**

The system is a standalone command-line application and does not have direct dependence on external or interfaces.

### **User Interfaces**

The system provides a command-line user interface for interaction. Users can input arithmetic expressions 3eeand receive the calculated results as output. The Command-line interface is designed to be intuitive and user-friendly, allowing users to utilize the system with ease.

### **Hardware Interfaces**

The system does not rely on any specific hardware infercases and is intended to be platform independent.

### **Software Interfaces**

The evaluator is implemented in C++ and does not require. integration with external software libraries or interfaces. It runs as a self-contained program.

### **Communication Interfaces**

This system does not involve a network or communication interface.

### **Memory Constraints**

The system is designed to be memory-efficient, but the exact memory constraints are subject to be dependent on the user’s device. But this program should be functional on any modern device.

### **Operations**

The primary operation of this system is to evaluate arithmetic expressions given by the user. This involves tokenization, operator evaluation, and result calculations. The system will also perform error handling for scenarios such as division by zero or any other invalid expressions.

## Product functions

The system core functionality includes:

* Parsing and evaluating arithmetic expressions
* Handing operator precedence and parentheses
* Recognizing calculating numeric constants.
* Providing informative error messages for invalid input.
* Supporting a variety of arithmetic operators.

## User characteristics

The primary users of the system are expected to have a basic understanding of the arithmetic expressions in use. Users may include students, programmers, and mathematicians who ended a quick tool or expression evaluation. The system's user interface is designed to be accessible and intuitive for both novice and experienced users.

## Constraints

The following constraints are considered during system development:

* The system must adhere to the mathematical rules for arithmetic expressions
* Division by zero is not allowed, and the system must handle this scenario flawlessly
* The system must must be written in C++ to align with the project requirements

## Assumptions and dependencies

* The system assumes that user will input valid arithmetic expressions that conform to mathematical conventions
* It will depend on the C++ programming language and standard libraries needed for development

## Known Issues and Limitations

[If known issues, limitations arise we will note them in this section]

# Specific Requirements

Within the Software Requirements Specifications (SRS), this section defines the specific software requirements essential for guiding the design and testing of the "Arithmetic Expression Evaluator in C++" project. These requirements are articulated in a manner that facilitates the system's designers in creating a solution aligned with these stipulations and enables testers to rigorously assess whether the system effectively meets these demands.

1. **Expression Parsing and Evaluation**: The system is mandated to efficiently parse and evaluate arithmetic expressions, encompassing tokenization, strict adherence to operator precedence based on the PEMDAS rules, and precise result generation.

2. **Handling Operator Precedence and Parentheses**: The system must implement and manage operator precedence as per the PEMDAS rules, effectively handling expressions within parentheses to ensure accurate order of operations.

3. **Numeric Constants Recognition and Calculation**: The system should possess the capability to identify and compute numeric constants within input expressions, inclusive of both integer and floating-point constants, while ensuring precision in calculations.

4. **Error Handling for Invalid Input**: Robust error handling is a critical requirement, requiring the system to generate informative error messages for situations like division by zero or invalid expressions, ensuring graceful handling of unexpected input.

5. **Support for Various Arithmetic Operators**: The system is expected to provide support for a comprehensive set of arithmetic operators, encompassing addition (+), subtraction (-), multiplication (\*), division (/), modulo (%), and exponentiation (^). These operators should be processed accurately within expressions.

These specific requirements play a pivotal role in shaping the functionality of the "Arithmetic Expression Evaluator in C++," providing clear guidelines to the development process, ultimately leading to the creation of a reliable and accurate system that fulfills its intended purpose effectively.

## Functionality

The Functionality section of this Software Requirements Specifications (SRS) serves as a comprehensive delineation of the core capabilities and features essential for the "Arithmetic Expression Evaluator in C++" system. Presented in a natural language style, this segment outlines the functional requirements necessary for the system's proper operation. These functional requirements include the various feature sets, capabilities, and security measures that the system must embody to fulfill its intended purpose. While the typical organization of this section is by features, alternative structures may be considered, offering flexibility in presentation. Additionally, if application development tools or modeling tools are employed in the development process, this section may reference their availability, indicating the location and naming of the tools used for capturing the system's functionality. In essence, the Functionality section forms the bedrock of the SRS, elucidating the fundamental operations and capabilities required for the system to successfully achieve its objectives.

### <Functional Requirement One>

Functional Requirement One dictates the system's capability to handle diverse arithmetic expressions encompassing operators such as addition, subtraction, multiplication, division, modulo, and exponentiation, while adhering to operator precedence rules (PEMDAS). It necessitates the recognition and calculation of numeric constants, both integer and floating-point, and mandates robust error handling for scenarios like division by zero or invalid input expressions. In essence, this requirement defines the system's core functionality for proficiently managing arithmetic operations, constants, and error scenarios

## Use-Case Specifications

Use-Case Specifications play a pivotal role in delineating the functional requirements of the "Arithmetic Expression Evaluator in C++" system, as well as certain non-functional requirements. They encapsulate the practical scenarios and interactions that the system must support, outlining the behavior and functionality expected from the system under various conditions. These specifications are instrumental in defining how users, whether novices or experts, interact with the system, including the steps they take, the expected outcomes, and the boundaries of the system's capabilities. Use-Case Specifications serve as a comprehensive guide for designers and testers, ensuring that the system aligns with the intended functionality and user experience throughout its development and testing phases

## Supplementary Requirements

Supplementary Requirements, as an integral part of the "Arithmetic Expression Evaluator in C++" project, serve as a repository for additional requirements that extend beyond the functional and non-functional aspects outlined in the use cases. These encompass various non-functional requirements and development constraints essential for the project's success. Supplementary Requirements capture specifics such as performance benchmarks, security considerations, system limitations, and any technical prerequisites that are not explicitly detailed in the use cases. They ensure that all critical project aspects are addressed comprehensively, thereby contributing to the project's holistic success and adherence to essential standards and constraints.

# Classification of Functional Requirements

| **Functionality** | **Type** |
| --- | --- |
| User Registration: Allows the user to create accounts | Essential |
| User Authentication: Implement a secure login system | Essential |
| Arithmetic Operations: Support basic operations | Essential |
| Order of Operations(PEMDAS): Follow correct evaluation | Essential |
| Parentheses Handling: Allow users to use parenthese | Essential |
| Input Validation: Validate user input for specified format | Essential |
| Result Display: Shows the result of the evaluated expression | Essential |
| Error Handling: Displays important error messages | Essential |
| Memory Functionality: Allows users to store and recall results | Desirable |
| Clean Interface: Design an intuitive and user-friendly UI | Desirable |
| Accessibility: Restrict access based on specific user authentication | Desirable |
| Test Cases: Develop a suite of test cases for validation | Optional |
| Compatibility: Ensure compatibility with other various platforms/software | Optional |

# Appendices

Appendices, within the context of the "Arithmetic Expression Evaluator in C++" project's Software Requirements Specifications (SRS), serve as supplementary sections that may contain additional information, such as diagrams, tables, or explanations. The SRS explicitly defines whether these appendices are to be considered as binding project requirements or if they primarily provide supplementary insights that enhance the understanding of the requirements without imposing specific obligations. By clarifying the role and relevance of the appendices, the SRS ensures transparency and clarity regarding their impact on the project's specifications