Team 1

Arithmetic Expression Evaluator
Software Requirements Specifications

Version <1.0>

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>
09/10/24	1.0	Initial editing of the document	Entire team
13/10/24	1.1	Added Use Case Design Diagram	Maxwell Phachanla

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

The purpose of the Software Requirement Specification is to detail all information related to the project's software and its requirements.It is meant to give a more in depth view of what is required for the project software-wise.

1.2 Scope

This Software Requirements Specifications describes the requirements to be used by the Arithmetic Expression Evaluator project during development. The plans as outlined in this document are based upon the product requirements as defined in the Vision Document.

1.3 Definitions, Acronyms, and Abbreviations

SRS - Software Requirements Specification

+ (addition)

- (subtraction)

* (multiplication)

/ (division)

% (modulo)

** (exponentiation)

1.4 References

- *Project Plan (Part One)*: Copy of 348 Project Plan - Group 1.docx
- *GitHub repository*
- *Meeting Log*

1.5 Overview

This Software Requirements Specification contains the following information:

Overall Description — summarizes the factors that affect the project and its requirements.

Specific Requirements — 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.

Classification of Functional Requirements — lists the essential, desirable, and optional functional requirements in order of importance.

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

2.1.2 User Interfaces

A user-friendly and legible command-line interface that allows users to enter expressions and displays the calculated results.

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

Visual Studio Code (Majority)
Vim (through PuTTY)

2.1.5 Communication Interfaces

2.1.6 Memory Constraints

64-bit Windows system.

2.1.7 Operations

2.2 Product functions

The program should be able to detect valid expressions on the command line, and compute them. It should be able to detect operators, like addition, subtraction, multiplication, division, modulo, and exponentiation, as well as unary operators. It should also be able to detect parentheses and compute the expressions within them before any others. It should also be able to detect invalid expressions.

2.3 User characteristics

EECS students/faculty/field

Math students/faculty/field

Science students/faculty/field

Any user in need of a calculator for simple expressions

2.4 Constraints

Project must be coded in C++

Must be completed and submitted by December 12th, 2024

2.5 Assumptions and dependencies

The ability to do/learn C++

Everyone is able to access the program

Ability to execute C++ code file

Access to command line interface

2.6 Requirements subsets

3. Specific Requirements

3.1 Functionality

3.1.1 Expression parsing and validation

The program will need to be able to parse through a given arithmetic expression, determining if the given expression/input is valid, and should be computed if so.

3.1.2 Operator Support

The program should support the following operators: Addition, Subtraction, Multiplication, Division, Modulo, and exponentiation.

3.1.3 Parenthesis Handling

The program should be able to handle parenthesis and use these in order to determine the order of evaluation.

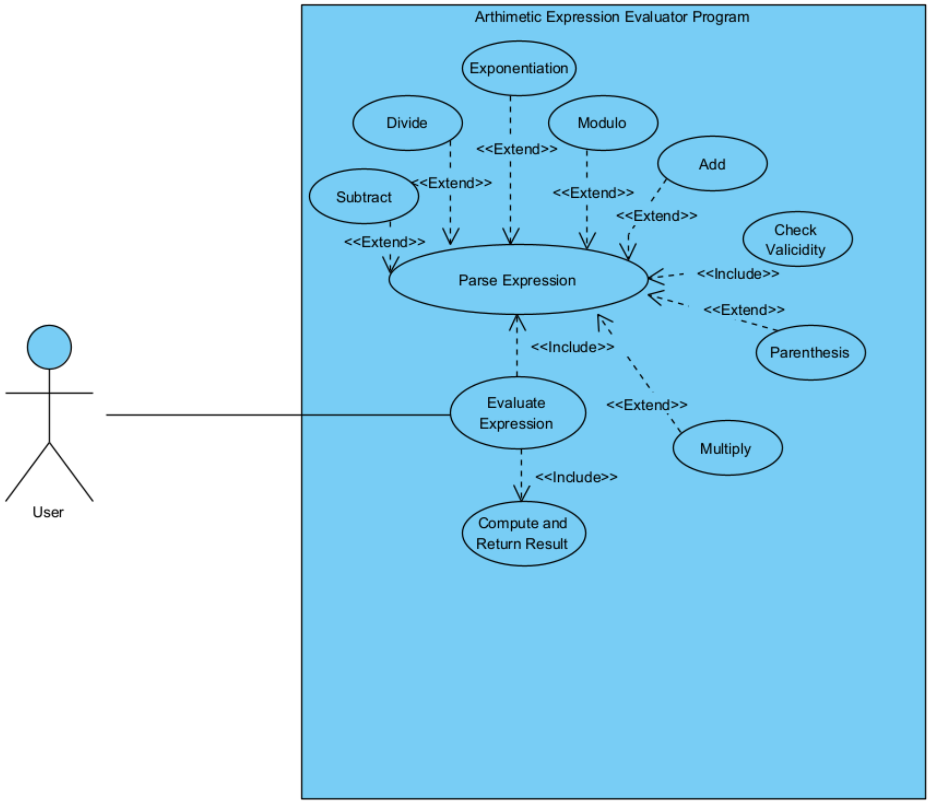
3.1.4 Numeric Constants

The program should be able to recognize and calculate numeric constants within the expression.

3.1.5 User-Friendly Interface

The program should be a user-friendly and legible command-line interface that allows users to enter expressions and displays the calculated results.

3.2 Use-Case Specifications



3.3 Supplementary Requirements

3.3.1 Fast Calculations

The program should return evaluation quickly, without taking excessive time.

3.3.2 Error Handling

The program should implement robust error handling to manage scenarios like division by zero or invalid expressions.
Simple and easy to use

4. Classification of Functional Requirements

Functionality	**Type**
Expression parsing and validation	Essential
Operator Support	Essential
Parenthesis Handling	Essential
Numeric Constants	Essential
User-Friendly Interface	Desirable
Error Handling	Desirable
Fast Calculations	Desirable

5. Appendices

Term Project Description