EECS 348 Group 16

Arithmetic Expression Calculator
Software Requirements Specifications

Version 1.0

Revision History

Date	Version	Description	Author
13/10/2023	1.0	Document Creation	Sam Muehlebach, Josh Welicky, Jennifer Aber, Jawad Ahsan, Basim Arshad, Mark Kitchin

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

The purpose of this document is to outline the expected behavior of the project, the Arithmetic Expression Calculator. This document will include the overall description of the requirements—including functional requirements, non-functional requirements and constraints—specific requirements, and a classification of the functional requirements to elucidate the software requirements of the project.

1.2 Scope

The focus of this document is the documentation of the requirements for the software product, which will evaluate arithmetic expressions and will handle operators +,-,*,/,% and ^, numeric operands, and matching parentheses. This document also contains the details of the use case specifications—in the form of a UML use case diagram—of the product.

1.3 Definitions, Acronyms, and Abbreviations

See the Glossary.

1.4 References

Project Description	—	Can be found in the EECS 348 Canvas page in the project file.
Glossary	—	Can be found in the Appendices section.

1.5 Overview

This *Software Requirements Specification* contains the following information:

Overall Description	—	describes the general factors that affect the product and its requirements. Includes information on interfaces with the product, product functions, user characteristics, constraints, and more.
Specific Requirements	—	describes the specific functional and supplementary requirements of the product. Also contains use case specifications in the form of a UML use case diagram.
Classification of Functional Requirements	—	lists all functional requirements, ordering them by the classifications: essential, desirable, and optional.
Appendices	—	lists the definitions of several terms used in this document.

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

Not applicable.

2.1.2 User Interfaces

This software will implement a command line interface (CLI), where users will input an arithmetic expression that satisfies the basic criteria for a valid expression, further outlined in the Project Description document. If the input is valid, the result will be displayed. If not, an error message will be displayed instead.

2.1.3 Hardware Interfaces

Not applicable.

2.1.4 Software Interfaces

This software will be compatible with all mainstream operating systems—such as Windows, macOS, and Linux—and will implement the <iostream>, <string> and <cmath> C++ libraries.

2.1.5 Communication Interfaces

Not applicable.

2.1.6 Memory Constraints

This software will not be on a scale large enough to warrant explicit memory constraints.

2.1.7 Operations

Not applicable.

2.2 Product functions

This product will be able to take in an arithmetic expression as an input. It will parse it and evaluate the expression according to the order of arithmetic operations GEMDAS. The product will obtain input and display output through a user-friendly command-line-interface. The product will be able to detect and respond to invalid arithmetic expressions.

2.3 User characteristics

Users of this product will be familiar with the concepts of basic algebra, such as the order of operations and the concepts of each arithmetic operation. There is no specific subset of individuals who will be using this product, such as students or employees.

2.4 Constraints

This product will be constructed in the C++ programming language using the principles of object-oriented programming. Documentation and comments detailing functionality and logic are required in the source code for this project. All project deliverables will be completed by December 7th, 2023. There is no budget or this project.

2.5 Assumptions and Dependencies

Assumptions: Valid user input, working hardware, compatible OS (Mac, Windows, Linux), and inputted numbers being standard integers or floating point numbers.
Dependencies: C++ compiling tool, access to standard C++ libraries (<iostream>, <string>, and <cmath>), access to a terminal to run program, and access to a supported operating system.

2.6 Requirements subsets

Not applicable.

3. Specific Requirements

3.1 Functionality

Arithmetic Expression Parsing

The system should be able to read an arithmetic expression input and parse it into smaller components. It should then be able to combine the results of the components to produce a final result.

Addition Operator

If the “+” operator is present in an expression, the system should be able to add the numeric constants before and after the “+”.

Subtraction Operator

If the “-” operator is present in an expression, the system should be able to subtract the numeric constant following the “-” from the one preceding it.

Multiplication Operator

If the “*” operator is present in an expression, the system should be able to multiply the numeric constants before and after the “*”.

Division Operator

If the “/” operator is present in an expression, the system should be able to divide the numeric constant preceding the “/” by the one following it.

Modulo Operator

If the “%” operator is present in an expression, the system should be able to take the modulus of the first operand with the second operand.

Exponentiation Operator

If the “^” operator is present in an expression, the system should be able to calculate the first operand to the power of the second.

Process Integer Constants

The system should be able to return and perform calculations on integer constants.

Process Decimal Constants

The system should be able to return and perform calculations on decimal constants.

Match Parentheses

The system should be able to read a user’s operation and recognize if all opening parentheses in the operation are matched with closing parentheses. If they are not, the system should prompt the user with an error.

Utilize Operator Precedence

The system should be able to recognize the priority of operators within an operation (PEMDAS) and evaluate the expression with respect to that priority.

Handle Zero Division

The system should be able to recognize if an operation contains a 0 or an internal operation equal to 0 when it follows “/” and if so, prompt the user with an error

Handle Invalid Expressions

The system should be able to recognize and properly respond to an input that includes characters that are not accepted or accounted for.

Process Unary Negation

The system should be able to recognize and calculate numbers that are negated.

Obtain Input

The system should be able to receive a user’s input properly and accurately for calculation through a command line interface.

Display Output

The system should be able to display the result of calculating a user's operation properly and accurately through a command line interface.

*** Operator*

If the "**" operator is present in an expression, the system should be able to calculate the operand to the left of the operator to the power of the operand to the right.

Match Brackets

The system should be able to read a user's operation and recognize if all opening brackets in the operations are matched with closing brackets. If they are not, the system should prompt the user with an error.

! Operator

If a "!" follows an operand, the system should be able to calculate the factorial of that operand.

Fractional Exponents

If a fraction proceeds the exponentiation operator ("^ or **"), the system should be able to perform the root of the function, of which depends on what the fraction is

Sine Operation

If "sin(x)" is in the operation, where x is an integer or an internal operation, the system should be able to calculate sine on that number.

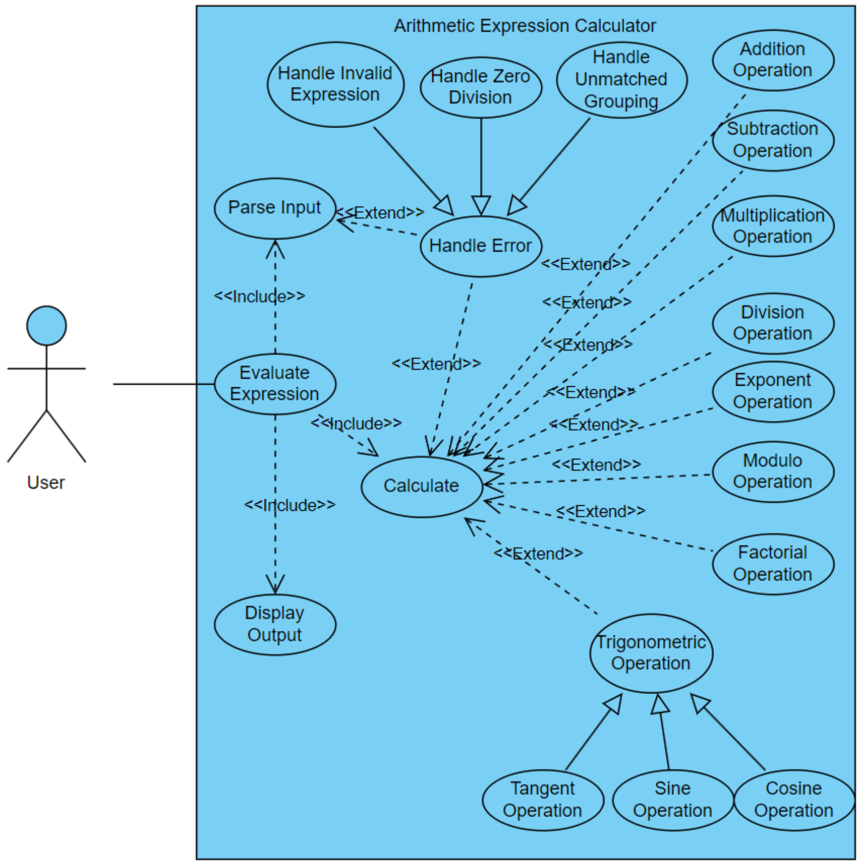
Cosine Operation

If "cos(x)" is in the operation, where x is an integer or an internal operation, the system should be able to calculate cosine on that number.

Tangent Operation

If "tan(x)" is in the operation, where x is an integer or an internal operation, the system should be able to calculate tangent on that number.

3.2 Use-Case Specifications



3.3 Supplementary Requirements

This product must be developed in the C++ programming language using data structures and object-oriented programming. This product, and all of its deliverables, must be completed by December 7th, 2023. This product should be compatible with all mainstream operating systems.

4. Classification of Functional Requirements

Functionality	Type
Integer Expression Parsing	Essential
+ Operator	Essential
- Operator	Essential

* Operator	Essential
/ Operator	Essential
% Operator	Essential
^ Operator	Essential
Process Integer Constants	Essential
Process Decimal Constants	Essential
Match Parentheses	Essential
Utilize Operator Precedence	Essential
Handle Zero Division	Essential
Handle Invalid Expressions	Essential
Process Unary Negation	Essential
Obtain Input	Essential
Display Output	Essential
** Operator	Desirable
Match Brackets ([])	Desirable
! Operator	Desirable
Fractional Exponents	Optional
Sine Operation	Optional
Cosine Operation	Optional
Tangent Operation	Optional

5. Appendices

Glossary:

GEMDAS: Acronym for Grouping, Exponent, Multiplication, Add, Subtract. Represents the order of operations when calculating a mathematical expression.

Parsing: The breakdown of a mathematical expression into its individual components.

PEMDAS: Acronym for Parentheses, Exponent, Multiplication, Add, Subtract. Represents the order of operations when calculating a mathematical expression.

Precedence: The order of operations within a given expression, or which parts of the expression are evaluated first. It can be thought of as an order of priority.

UML: An abbreviation for Unified Modeling Language. A visual language used for modeling, specification, and documentation. Used commonly in software engineering.

Unary Negation: Use of a minus sign (-) preceding an integer to indicate a value less than zero

Use Case Diagram: A UML diagram that is commonly used to discover and document the functional requirements of a software solution. A use case diagram maps the interactions between external entities and the system, as well as the purpose and results of those interactions.

***:* The symbol that can represent the exponential operation as a desirable feature.

%: The symbol used in C for the modulus, or remainder, operation.

!: The symbol representing the factorial operation. The factorial of an integer n is the result of the following expression: n! = (n) * (n-1) *…*1.