SN^{2}ARK CO.

Arithmetic Parser
Software Requirements Specifications

Version <1.1>

Revision History

Date	Version	Description	Author
10/07/23	1.0	Team members completed detailed descriptions for Part 2: The Software Requirements	Sophia Jacob, Reeny Hunag, Navya Nittala, Kusuma Murthy, Anna Lin, Nimra Syed
10/13/23	1.1	Team members revised the detailed descriptions of functionalities in Section 4	Sophia Jacob, Reeny Huang, Anna Lin, Nimra Syed

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

The purpose of this project is for SN^{2}ARK to develop a multifunctional arithmetic expression evaluator utilizing C++. The objective is to create a simulation of a calculator, through a program, that can handle expressions as input and calculate the result using mathematical operations like (PEMDAS). Furthermore, SN^{2}ARK will fully integrate the Software Development Process through timely deliverables within the semester. These deliverables include a detailed project plan, a requirements document, a design document that is in alignment with the requirements, a set of test cases, and a fully developed product. As part of the scope of this project, each iteration should be timely and in line with the functionality of the requirements.

1.2 Scope

This ***Software Development Plan*** is in association with the overarching plan of the Arithmetic Parser project. It encapsulates the management process, objectives, development, role specification, the iterations within the project, and detailed accounts of each iteration described in the Iteration Plan.

The plans as defined in the document are based upon the project requirements as outlined in the *Requirements Document*, mentioned on Canvas.

1.3 Definitions, Acronyms, and Abbreviations

See the Project Glossary.

1.4 References

For the ***Software Requirements Specification***, the list of referenced artifacts includes

1. 001 Project Plan, SN^{2}ARK, 24/09/23
2. Iteration Plans
3. Project Management Plan
4. Requirements Document
5. Design Document
6. Test Cases
7. 007 Glossary, SN^{2}ARK, 24/09/23

1.5 Overview

This ***Software Requirements Specifications*** document contains the following information:

Overall Description – Provides an overview of the project's purpose, scope, and objectives. It also defines the deliverables that the project is expected to complete. In addition, this document is a comprehensive guide to display the specific requirements that are imperative for the Arithmetic Parser.

Specific Requirements – The SRS outlines the functional and non-functional requirements that are necessary for the full completion of the Project.

Classification of Functional Requirements – Explains the different types of requirements scoped into the Arithmetic Parser, including the essential, desirable, and optional requirements.

2. Overall Description

2.1 Product Perspective

The Product Perspective is to create a functional arithmetic parser, with expectations of large or small math equations inputted and correct outputs displayed.

2.1.1 System Interfaces - N/A

2.1.2 User Interfaces

The user interface is a GUI consisting of input for the user, including a variety of symbols and operations, which they can use to construct Mathematical Equations. These can then be inputted for the Arithmetic Parser to display the correct output.

2.1.3 Hardware Interfaces - N/A

2.1.4 Software Interfaces

The Software Interfaces should be consistent with Git and GitHub, and the team members will use C++ to complete this Software Project. The project should be compatible in a Linux environment. SN^{2}ARK will use Visual Studio Code, to contribute equally to the project. Additionally, the Software Interface will handle errors. Team members will also use Use Case Models and Class Diagrams to outline the approach to creating the Arithmetic Parser as part of their Software Interfaces.

2.1.5 Communication Interfaces - N/A

2.1.6 Memory Constraints - N/A

2.1.7 Operations - N/A

2.2 Product functions

The product functions should perform the functionalities of a calculator, and assist users with mathematical equations that they would like to solve.

2.3 User characteristics

The users include individuals affiliated with EECS 348, specifically the TA, Professors, and students. All users have interacted with a calculator, and have elementary math skills. This will play a part in the development of the project, as users can find the right answer to their math equation, and the Arithmetic Parser must consistently present accurate results. Also, since users have used calculators before, the GUI should be readable and easy to use in comparison.

2.4 Constraints

The constraints consist of completing the project before 5 December 2023, developed using C++, and within the Linux environment.

2.5 Assumptions and dependencies

Team members assume that users have a laptop where they can interact with the Arithmetic Parser to generate results. It is also assumed that the users have elementary math skills, as defined before. The requirements also depend on the participation of each team member to ensure requirements are done efficiently and on schedule.

2.6 Requirements subsets- N/A

3. Specific Requirements

3.1 Functionality

3.1.1 Expression Parsing

The program should be able to parse arithmetic expressions entered, ensuring operator precedence and parenthesis rules are followed.

3.1.2 Operator Support

The Arithmetic Parser allows for these essential following operators:

- Integer Operands and the + Operators

- Program can parse through integers and addition operands.
- The Minus Operator
  - Program can parse through minus operands.
- The * Operator
  - Program can parse through the multiplication operator.
- The / Operator
  - Program can parse through the division operator.
- The % Operator
  - Program can parse through the modulo operator.
- The ^ Operator
  - Program can parse through the exponential operator.

3.1.3 Parenthesis Handling

The Arithmetic Parser ensures that the program can handle expressions with single or multiple expressions enclosed with parentheses to determine the order of evaluation. Parentheses for grouping should be allowed.

3.1.4 Numeric Constants

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

3.2 Use-Case Specifications

Functional Requirements of System:

Use Case: Calculate

- Description: A software user interface that performs and displays arithmetic calculations.
- Pre-condition: The User inputs in arithmetic operations.
- Post-condition: Software displays correct computation of the arithmetic express that was inputted.
- Basic path: The user enters a valid expression and our software/calculator outputs the correct answer.
- Alternative Path: The user enters an invalid expression and our calculator outputs an error message.

Includes of Use Case: Calculate

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Division)
- % (Modulo)
- ^ (Exponentiation)

Non-Functional Requirements of System:

- **Expression Parsing:** There must be a function implemented that tokenizes the input expression. Additionally, the program must contain a data structure that represents the expression's structure.
- **Operator Precedence:** Implement the logic that evaluates the expression according to operator precedence.
- **Parenthesis Handling:** The program must have a mechanism that identifies expressions within parentheses, and then evaluates it.
- **Numeric Constants:** The program should recognize numeric constants in the input.
- **User Interface:** The program should be user-friendly and have a readable command-line interface that allows users to input their expressions, and receive desirable calculated results.

3.3 Supplementary Requirements

Other Requirements:

- **Performance/Response Time**: The program should be able to provide immediate output depending on the input provided by the user.
- **Data Storage:** The program may require temporary data storage for the calculations while evaluating the expressions.
- **Maintenance:** Following each iteration, regression testing must be implemented, to ensure that all previous functionality is intact with the new addition. Additional testing and debugging may be conducted to ensure the proper functionality of the program.

4. Classification of Functional Requirements

Functionality	**Type**
Software Shell has the functionality to take in two numerical values to do addition.	Essential
Software Shell has the functionality to take in two numerical values to do subtraction.	Essential
Software Shell has the functionality to take in two numerical values to do multiplication	Essential
Software Shell has the functionality to take in two numerical values to do modulo.	Essential
Software Shell has the functionality to recognize numeric constants and perform calculations with them.	Essential
Software Shell has the ability to take in an expression and follow orders of operation and parenthesis hierarchy to do expression parsing.	Essential
Software Shell has the functionality to perform exponentials on a numerical value.	Desirable
Software Shell has the functionality to allow operations (addition, subtraction, multiplication, and exponents) for Floating Point Numbers, if the user desires.	Desirable
Software Shell has the functionality to recognize mathematical brackets compared to parenthesis and follow the order of expressions based on their hierarchy.	Optional
Software Shell has the functionality to recognize mathematical commas and follow the order of expression.	Optional
Software Shell provided buttons of operations for users to click.	Optional

5. Appendices

Not applicable at the current stage of the project. Refer to 001 - Project Plan for any additional information on the scope and definition of the project.