17 Co. Calculator
Software Requirements Specifications

Version 1.0

Revision History

Date	Version	Description	Author
12/Oct/23	1.0	Initial Software Requirements Spec	*17 Co.* Team

Table of Contents

Software Requirements Specifications

1. Introduction

This Software Requirements Specifications (SRS) document serves as a detailed description of the requirements engineering phase of 17 Co. Calculator. It captures the complete requirements of the entire project, with more details in section 1.2, entitled Scope.

Reading each section will leave any reader with a detailed understanding of the requirements of 17 Co. Calculator. This is highly important so as to reduce any chance of miscommunication or misunderstanding while future phases (as references in the Software Development Plan (SDP)) are developed.

This document is being written as part of Phase 2; however, future iterations may have changes, and those changes are tracked by version number. If a new phase requires change to this document, the version will increment by one (ex: 1.0 -> 1.1). If a change is needed within a phase, an additional number is to be added and incremented (ex: 1.0 -> 1.0.1). A detailed log of changes is also to be kept on page 2 of the document.

1.1 Purpose

The purpose of the SRS is to describe the functionality and behavior of 17. Co Calculator both as a closed system and any possible connections to external systems, although none should be needed. This document also describes the nonfunctional requirements, such as ease of use and portability, and constraints imposed by shareholders and artifacts.

The end goal is that, after reading the SRS, anyone can understand the factors and functions needed to complete the 17. Co Calculator.

1.2 Scope

The SRS applies to the entirety of the 17 Co. Calculator and may have sections that refer to more specific sections within. However, the entirety of this document describes the purpose, features, and requirements of the calculator as a whole. This means that the document applies to all use-cases, and all other aspects of the development of 17 Co. Calculator.

1.3 Definitions, Acronyms, and Abbreviations

Various terms, acronyms, and abbreviations may be used within the Software Requirements Specification. A complete list of important ones can be found here:

SRS: Software Requirements Specification. This document.

SDP: Software Development Plan (see references for more details)

PEMDAS: The order of operations. Which is: Parenthesis, then Exponents, then Multiplication and Division, and then finally Addition and Subtraction.

Other terms may be used that are within the project glossary, with file ID D00002.

1.4 References

Various sections of the SRS may reference other documents. You can see a list of all of them here:

PROJECT GLOSSARY – File ID D00002

SOFTWARE DEVELOPMENT PLAN – File ID D00001

1.5 Overview

The document is broken up into various sections:

Section 1: Outlines the high-level details of the SRS, and gives context to the document within the project as a whole
Section 2: Outlines the overall description of the project and what the end goals of the project are.
Section 3: Outlines the requirements of the project in order to meet the end goals described in section 2.
Section 4: Outlines the entirety of the requirements and classifies them by their requirement type
Section 5: Outlines supplementary material that can help bolster an understanding of the requirements of the project. There are currently no appendices, although future iterations of the SRS may include some.

Each section is broken up into various subsections to clarify points further. See the table of contents above to navigate subsections.

2. Overall Description

The primary goal of this software project centers on the development of a highly functional calculator, driven by a commitment to practicality and dependability. At its core, this calculator aims to provide users with a powerful, consistent, and reliable tool for executing a wide range of mathematical calculations. Rather than venturing into the realm of revolutionary innovation, our primary objective is to create a solution that users can consistently rely on when tackling mathematical computations and similar tasks.

2.1 Product perspective

This product has a somewhat unique perspective, by prioritizing practicality over revolutionary features, we aim to meet the core mathematical needs of our users efficiently. This product perspective emphasizes precision, reliability, and user-friendliness as the key pillars that underpin the calculator's development. Ultimately, this approach ensures that our software becomes an essential companion for individuals in various fields, simplifying their mathematical endeavors and enhancing their productivity.

2.1.1 System Interfaces

The system needs to interface with both the computer's hardware and the software application that is being run. It needs to interact with the software to perform the operations in a specified way, determined by the written software. The system also must interface with the hardware, as the hardware of the computer being used is required to run the program and get it to perform as intended. For example, the keyboard must be used for the user to enter any mathematical expressions they want the program to evaluate. The display must be used to show the user what they have typed in as their expression and to show the result of the evaluated expressions. The CPU is interfaced with as the program needs to be run by the computer and perform various mathematical calculations. The memory is needed to store the expressions that are given by the user and the instructions for determining how the expression should be evaluated, and the rules of performing the operations.

2.1.2 User Interfaces

The user interface needs to allow the user to input any expression using the available operators. It must contain an area for the user to type in their expression and any necessary operator symbols, such as addition, subtraction, multiplication, parenthesis, unary operators, etc. The user interface must also allow for the user to see the result of any expression given to the program.

2.1.3 Hardware Interfaces

The hardware interface must allow for the program to recognize inputs from the keyboard and enter them into the program. The display being used must be able to show the expression the user is typing in as they are typing it. The display also needs to be able to show the result of the expression after the user requests for the solution or display that an invalid expression has been entered.

2.1.4 Software Interfaces

The software must interface with the user. The mathematical expressions given by the user must be able to be interpreted by the software to be evaluated. The software must be able to interpret the expression and evaluate whether it can be solved, and if so, it evaluates the solution and returns it to the user.

2.1.5 Communication Interfaces

The program needs to be able to communicate with the user when an invalid expression is attempted to be calculated. The program must display that the given expression is invalid to communicate that with the user. If the user inputs a valid expression, then the program computes the solution and visually displays it to communicate the result to the user.

2.1.6 Memory Constraints

The program must not contain any memory leaks. It should be optimized so it uses as little memory as possible without compromising the performance and accuracy of the program.

2.1.7 Operations

The operations that the program must be able to compute include, addition (+), subtraction (-), multiplication (x), division (/), parenthesis (), exponents (^), unary operators (-6). These should be evaluated in the proper order of operations, or PEMDAS.

2.2 Product functions

The primary, general function of this project is to evaluate a user-entered mathematical expression. As such, all the functions we will specify later will be in service to this goal. In brief, it must be capable of evaluating at least more basic expressions according to PEMDAS and other basic mathematical rules. It must also complete other basic software requirements (such as error handling) since it is a software product.

2.3 User characteristics

Those who use this project will likely be a more general audience. Therefore, they will not be heavily technical and as such we must design this product with ease of use and other such characteristics in mind. There are also a variety of other calculator options publicly available and so this product should be different enough to attract attention while also being similar enough to satisfy user expectations.

2.4 Constraints

There are a variety of constraints that affect our project. We must use object-oriented programming methodology to design our project. We must include comments in our code and documentation of our code to explain the logic and functionality behind it. We must also test our code by developing unit tests to make sure our project works, and we must make sure that we provide helpful error messages to the user if they enter invalid input. In general, we must follow proper software engineering methodologies and ensure the completion of project deliverables by stated deadlines.

2.5 Assumptions and dependencies

Our project will depend on both the C standard input-output library <stdio.h> and the standard math library <math.h>. We will also require a C compiler to compile our program, as well as tools such as GitHub to host our code and collaborate, Canvas and Outlook for messaging and information sharing and communication, a terminal and operating system to run our program, and an IDE, Visual Studio Code, to write our work.

We will assume the user is not technically literate beyond being able to run our program, but that they understand the common mathematical notion that we will use (such as parenthesis to separate expressions, + for addition, etc.)

2.6 Requirements subsets

There are a few main subsets of our requirements. We have many functional requirements as to what features our users would expect, and we have many non-functional requirements that help specify how our program should function on a broader level, beyond any one particular feature. They often describe how the product experience should be as a whole.

3. Specific Requirements

This section of the document will contain all the requirements our product will satisfy. It will have enough detail that systems will be able to be designed to satisfy them and so that the requirements are testable to ensure a successful outcome.

3.1 Functionality

This section of the document will define all of the functional requirements our product will need to satisfy in order to succeed. This section can be thought of as including all major functional features our product will need to have.

3.1.1 Addition

The addition functionality means it should be able to accept any two valid data objects and return their sum as another valid data object.

3.1.2 Subtraction

The subtraction functionality means it should be able to accept any two valid data objects and return their difference as another valid data object.

3.1.3 Parenthesis

The parenthesis functionality means it will need to be able to accept a mathematical expression, scan for the outermost set of parentheses, and call the normal functions on the expression within said parenthesis before returning the simplified expression. This should also cover cases of extraneous parenthesis.

3.1.4 Multiplication

The multiplication functionality means it will need to accept two valid data objects and return the product of those two objects as another valid data object.

3.1.5 Division

The division functionality means it will need to accept two valid data objects and return the quotient of those two objects as another valid data object.

3.1.6 Exponentiation

The exponentiation functionality means it will need to accept two valid data objects and return another valid data object. The resulting data object should be the result of the 1^{st} to the power of the 2^{nd}.

3.1.7 Mixed Operators

The mixed operator functionality means it should be able to accept a mathematical function with potentially ambiguous operators and appropriately call the functions in accordance with proper order operations in mind (PEMDAS).

3.1.8 Unary Operators

The unary operators functionality means it should be able to accept certain operators that are only acting upon a single number i.e.. -(-8) ==> 8.

3.2 Use-Case Specifications

Functional Requirements:

Addition: The addition functionality will take the two numbers given by the user and add them together, producing a sum.

Subtraction: The subtraction functionality will take the two numbers given by the user and subtract the second number and subtract it from the first, producing a difference.

Multiplication: The multiplication functionality will take the two numbers given by the user and multiply them together, producing a product.

Standard Division: The division functionality will take in the two numbers given by the user and will divide the first number by the second number to get a quotient.

Modulo Division: The modulo division functionality will take in the two numbers given by the user and will divide the first number by the second number, producing a quotient. Then, the functionality will calculate the remainder of the quotient, producing a remainder quotient.

Exponentiation: The exponentiation functionality will be able to multiply any number of numbers by itself as many times as the user requires. This is important in long sequences of any of the previous functionalities if they need to be multiplied by itself some number of times.

Parenthesis Recognition: The calculator will be able to understand the proper way of solving complex equations involving parenthesis through order of operations solving techniques.

Numeric Constants: The calculator will also be able to recognize and run operations on only numeric constants. If any input breaks numeric constant convention, the calculator will recognize the error and handle it.

Non-functional Requirements:

Easy Use: The calculator should be simple upon launch and user friendly. A simplistic design will be used to ensure that users of all levels, from beginner to advanced, can navigate, and utilize the calculator's functionality.

Performance: The calculator should be built effectively with input and output speed being quick.

Security: The calculator should be secure with the user only having access to what is presented in the interface.

Portability: The calculator should be able to port to different devices and work just as effectively.

3.3 Supplementary Requirements

Error Catching:

Invalid Input: If the system detects an invalid input such as letters, it should not crash, rather it should raise an Invalid Input Error exception that the user would have to clear from the screen for them to continue using the calculator.

Arithmetic Errors: Any error that is deemed an arithmetic error such as division by zero, overflow (too large of a number to be represented) or underflow (too small of a number to represented) will also be handled by raising an Error (Overflow, Underflow, Division by Zero, etc.) exception that the user must clear to continue using the calculator

Syntax Errors: Any error where the user enters improper syntax, and it results in the calculator being unable to operate should raise a Syntax Error exception that tells the user their formatting for the equation is incorrect. Errors like mismatched parentheses and incomplete expressions fall under the syntax error category.

Function Errors: If an error is raised during a function call will raise a Function Error Exception. Function

errors could be arithmetic errors pertaining to certain functions such as passing in a negative number into the square root function.

4. Classification of Functional Requirements

Functionality	Type
Addition	Essential
Subtraction	Essential
Parenthesis	Essential
Multiplication	Essential
Division	Essential
Exponentiation	Essential
Mixed Operators	Essential
Unary Operators	Essential
Easy to Use	Desired
Performance	Desired
Portability	Desired
Security	Desired
Error Handling	Desired
Advanced Mathematical Functions	Optional
Memory of Previous Calculations	Optional
Multi-layered Button Layout (multiple functionalities assigned to a single button) (Ex: 2^{nd})	Optional

5. Appendices

There are no appendices as of SRS v1.0.