<Apex Software>

<Tech Calculator>
Software Requirements Specifications

Version <1.2>

Revision History

Date	Version	Description	Author
<10/10/24>	<1.0>	<Initial drafting of SRS>	<Gage Weaver>
<10/11/24>	<1.1>	<Further drafting of SRS>	<Gage Weaver>
<10/16/24>	<1.2>	<Section 3.2 completion, and changes to 3.3 to provide concise descriptions>	<Gage Weaver>

Table of Contents

Software Requirements Specifications

1. Introduction

The software Requirements Specification will contain requirements that the finished tech calculator will need to have. Additionally it will include a use case model for the application, and any possible performance requirements the project may have. This document will be referenced to ensure that development includes all requirements outlined in the SRS

1.1 Purpose

The purpose of this software is to provide a working tech calculator which accepts user input and outputs correct info, which will be done by parsing different expressions, checking for parentheses, and handling errors. Ideally, the calculator would respond in a relatively quick time, in a user-friendly interface along with a user manual to provide additional guidance. C++ will be the language used for all backend development.

1.2 Scope

This document will apply only to the Tech Calculator application developed by Apex Software. Tech Calculator is an application that will take arithmetic user input and perform calculations following PEMDAS order of operations. It is associated with the Tech Calculator use case model, which is present in section 3.1 and 3.2 of this document.

1.3 Definitions, Acronyms, and Abbreviations

PEMDAS: An acronym describing the arithmetic order of operations. They are to be completed in the following order: Parentheses, exponents, multiply & divide (left to right precedence), add & subtract.

1.4 References

N/A

1.5 Overview

The remainder of this document will contain a brief description of the different possible uses of tech calculator that could affect its requirements. After that in section three of the document will be the more specific requirements for the project including its functionality and use-case specifications. Lastly there is a classification of the functional requirements of the application based on their priority.

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

Not applicable.

2.1.2 User Interfaces

Ensure the calculator functionality is accessible and intuitive at its basic level, all other functionality should be outlined in detail in the user manual.

2.1.3 Hardware Interfaces

Not applicable.

2.1.4 Software Interfaces

The front end interface will be able to interact with the backend interface to be able to make function calls to evaluate expressions.

2.1.5 Communication Interfaces

Not applicable.

2.1.6 Memory Constraints

Memory constraints for a calculator should not be an issue on modern computers, however limiters will be put into place to limit excessively large input equations

2.1.7 Operations

Not applicable.

2.2 Product functions

The product will function as an arithmetic calculator and be able to add, subtract, multiply, divide, modulus operator, and parse parentheses. Along with handling bad input.

2.3 User characteristics

The user will be a human person that has access to a computer and would like to evaluate expressions.

2.4 Constraints

Constraints for the calculator is it must be developed in C++ and it should be able to run on most modern computers. Additionally there will be a maximum safe input length that will be determined at a future stage of development.

2.5 Assumptions and dependencies

Assumptions made is that the user will know how to type into an interface to be able to input expressions into the calculator. No dependencies except for the user being able to run an exe file.

2.6 Requirements subsets

Not applicable.

3. Specific Requirements

3.1 Functionality

3.1.1 Expression Parsing

This calculator program will correctly evaluate arithmetic expressions. There is the option for the user to evaluate multiple inputs that can create the same mathematical expression and output, and these must be parsed correctly in every scenario. The calculator will follow PEMDAS order of operations for expression ordering. This can be verified by hand testing results and known working online calculators with test equations.

3.1.2 Arithmetic Operations

The calculator will correctly perform the arithmetic operations in PEMDAS along with the modulus operator. It will return consistent results for identical expressions and results can be verified by other calculators and hand arithmetic

3.1.3 User Input

The calculator will accept accurate user input from a keyboard. It will allow them to change expressions after input with the backspace key. The User will have enough space to type their entire expression. This can be tested by inputting an expression at the maximum length and ensuring that it is formatted correctly.

3.1.4 Error Handling

If the user attempts to input an invalid expression, such as dividing by zero, the program should correctly handle this error. This type of input should not result in the program crashing, and should instead display an easily readable message so that the user may correct their input. Another example of an invalid input would be an uneven amount of open and closed parenthesis. This can be tested by purposefully inputting all errors that are to be handled as specified in the user manual, including divide by 0, uneven parentheses, and sequential operators.

3.2 Use-Case Specifications

Description: The following use case specification will be encapsulating the functionality of the tech calculator, which allows a user to perform the arithmetic operations as specified in PEMDAS, along with the modulus operator.

Actors: There is one actor associated with this use case, that being the user who launches the tech calculator application and uses the calculator.

Preconditions: The user launches the tech calculator application

Postconditions: The calculator either displays the result of the expression entered to the user, or provides them with a meaningful error as to why the expression could not be evaluated. Then allows the user to input another expression

Event Flow:

-Main:

Step 1: The user will input a valid expression using their keyboard

Step 2: The user will press enter to evaluate the expression and will receive a result

Step 3: Repeat steps 1-2 until an invalid expression is entered (in which case the user is now in the alternate flow) or the calculator is exited out by the user

-Alternate:

Step 1: The user will input an invalid expression using their keyboard (arithmetic error, invalid characters etc.)

Step 2: The calculator will display an error that the expression could not be processed and allow the user to enter a different expression

Step 3: Repeat steps 1-2 until a valid expression is entered (in which case the user is now in the main flow) or the calculator is exited out by the user

Assumptions:

-The user knows how to use a keyboard, and has an understanding of basic arithmetic expressions and rules.

-The user has a device that can run the tech calculator application.

3.3 Supplementary Requirements

Nonfunctional requirements:

Reasonable Calculation Time: The calculator will evaluate and return a result for expressions in less than 1 second.

Interface: The calculator's interface should be simple and straightforward enough such that the user will not question how to use the calculator after reading the user manual.

4. Classification of Functional Requirements

Functionality	Type
Expression Parsing	Essential
Arithmetic Operations	Essential
User Input	Essential
Error Handling (ex. /0 or invalid expressions)	Essential
Consistent Input/Output Formatting	Desirable

5. Appendices

N/A