KU EECS 348: Software Engineering 1

Code Commanders
Software Requirements Specifications

Version 1.0

Revision History

Date	Version	Description	Author
03/24/2024	1.0	Initial revision	Code Commanders

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

This document will explain the functional/nonfunctional requirements of the Boolean calculator project and any other requirements and constraints needed to ensure the program functions expectedly. This document will address all the functions needed to ensure this is readily available and usable by the user and will be stated during the development stages to ensure that the project runs smoothly.

1.2 Scope

The document will describe the requirements for the calculator and any possible external interfaces. It will include functional requirements that are fundamental to the program, non-functional requirements that will improve performance on the user and constraints, which are set by the core project requirements for EECS 348 and the ability to run on an appropriate UI.

1.3 Definitions, Acronyms, and Abbreviations

The document will use the same terminology described in the project plan.

1.4 References

The document will reference the project plan for the appropriate definitions and the use case and class diagrams provided in the appropriate file.

1.5 Overview

This document will explain the functional/nonfunctional requirements of the Boolean calculator project and any other requirements and constraints needed to ensure the program functions expectedly. This document will address all the functions needed to ensure this is readily available and usable by the user. It will also provide the specific requirements alongside small explanations of each requirement, which are both functional and non-functional. The constraints are also explained here to understand where they come from

In addition, the functional requirements are determined as *necessary, desirable, or optional*. Necessary requirements are required for the function to operate as any other calculator would and are. Desirable requirements are used in addition to the necessary requirements to fulfill our goal of developing Boolean calculator. Optional requirements extend from the basic functions but can be removed or omitted without affecting the main functionality.

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

N/A

2.1.2 User Interfaces

We will design the user interface with the following attributes to ensure a quality user experience and ease of use, any optional components will be marked as such:

- Clear and defined questions with instructions to gather user input
- Clear and defined option choices to guide the user to make correct choices
- Error handling to capture and provide feedback for all possible choices by the user, even if the inputs are incorrect/unacceptable
- Sleek and uncluttered design
- (Optional) Online User Interface with program being held on a server and always accessible

2.1.3 Hardware Interfaces

This program does not require any physical hardware for this to function

2.1.4 Software Interfaces

Our program will be coded in C++ meaning that it will have an executable file that can be run to pull up a console where the user can access the UI. We may also implement an online UI with our code being run on an online server, so our application does not need to be downloaded.

2.1.5 Communication Interfaces

N/A

2.1.6 Memory Constraints

There are currently no memory constraints defined for this project, though throughout the project we will restructure code to be as memory efficient as possible.

2.1.7 Operations

N/A

2.2 Product functions

The Boolean Calculator simulates the behavior of logic circuits, including operations such as AND, OR, NOT, NAND, and XOR. The program will be able to handle complex logic circuits with multiple gates and input/output signals. We aim to have our calculator also output a truth table for the input function as well, but this function is not required and is optional.

2.3 User characteristics

The primary users of the Boolean Calculator are students and educators in computer science. Users are expected to have a moderate level of technical proficiency in software operations and a basic understanding of Boolean operations. An understanding of C++ is not required for end-users but is useful in understanding or modifying the source code.

2.4 Constraints

The Boolean Calculator must be developed in C++. This constraint limits the choice of libraries and development tools to those compatible with C++ environments. The calculator must be optimized for performance and capable of evaluating Boolean expressions without significant delay. Efficient memory usage will be required for handling large or complex expressions. The development process is constrained by the academic schedule for the spring 2024 term.

2.5 Assumptions and dependencies

The project assumes availability and permission to use certain C++ libraries or tools that facilitate parsing, expression evaluation, and user interface development. It is assumed that users will have access to an environment capable of running C++ applications. Dependencies on specific operating systems or hardware capabilities are minimized.

Requirements subsets

N/A

3. Specific Requirements

3.1 Functionality

Functional Requirement One: ***Operation Support***

- AND (&): Returns True if both operands are True
- OR (||): Returns True if one of the two operands are True
- NOT (!): Inverts the operand from TRUE to FALSE or FALSE to TRUE
- NAND (@): Opposite of AND, Returns FALSE if both are TRUE or Returns TRUE if both are FALSE
- XOR ($): Returns TRUE if exactly one operand is TRUE

Functional Requirement Two: ***Expression Parsing***

- Using infix notation we will parse through a Boolean expression.
  Ex: (T + F) * F

Functional Requirement Three: ***Truth Value Input***

- We will allow the user to enter any form of Boolean expression they wish with T for true, F for false.
  || - OR
  &- AND ...

Functional Requirement Four: ***Evaluation and Output***

- We will calculate the final value of the equation and output TRUE OR FALSE.

Functional Requirement Two: ***Error Handling***

- If user inputs incorrect: parentheses, operands, expressions, we will have correct error messages to allow the user to re-enter the expression again with the correct format.

3.2 Use-Case Specifications

1. User is welcomed with an intro page that says “Enter a Boolean expression to evaluate or enter ‘q’ to quit program:”		
2. User enters a Boolean expression		Alt. Flow 2. 1. User Enters ‘Q’
3. The program parses the Boolean expression		2. Program Terminates “Goodbye Partner
4. The expression is deemed valid.	Alt Flow 4. The expression is deemed invalid.	
5. The program evaluates the expression.	5. The program outputs an error message to the user stating the error and reason.	
6. The program outputs ‘TRUE’ or ‘FALSE’ and the truth table for the expression.		

3.3 Supplementary Requirements

Nonfunctional requirements will include but are not limited to:

-A provided README file that efficiently and accurately guides the user on how to use the program and contains explanations and examples of each of the different operators.

-The program text and output will be formatted uniformly and plainly to ensure the program is easy to read and use.

-The program will be able to handle long and complex Boolean expressions without raising errors or crashing.

-In the event of an error, a descriptive error message must be output to the user.

-During the parse of the expression, if the expression is deemed invalid the program will record the reason(s) for use during error production.

-In parsing the expression, the program will store sub expressions within parentheses to be used in the creation of the truth table. Ex (T||F) from (T||F)&F

Additionally, the development constraints are as such:
-The program must be implemented in C++

4. Classification of Functional Requirements

Functionality	Type
Perform all the possible Boolean Algebra functions (operator support)	Essential
Handle multiple order of operation functions appropriately	Essential
Output a truth table for each expression	Desirable
Design an online UI	Optional

5. Appendices