Boolean Logical Simulator
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
10/04/2024	1.0	Discusses the architectural design and methodology of the project statement in detail	<name>

Table of Contents

Software Architecture Document

1. Introduction

This document provides the architectural overview of the Boolean Logic Calculator as well as more specific details about the architecture. It will go over the scope, purpose, definitions, and decisions involved in the process.

1.1 Purpose

The purpose of this document is to discuss the basic design and architecture behind the Boolean logic simulator, as well as the programming constraints and use case values implemented.

1.2 Scope

This document covers the design of the algorithm followed to complete the Boolean Logic Simulator, covering software used for both backend and frontend development, as well as the constraints for the same. Thus simulator encompasses calculations of the Boolean expressions which has been explained in Software requirements.

1.3 Definitions, Acronyms, and Abbreviations

Software Architecture: The set of design decisions made during the development process about a system/software to be developed.

Object-Oriented Design: A design paradigm focused around designing separate functionalities in their own modules/objects, with each module having its own responsibilities.

1.4 References

This document references the Project Plan for definitions and guidelines as well as the Software Requirements Specifications for the functionalities and constraints of the software being developed.

1.5 Overview

This document is structured to provide an Introduction, Architectural representation, goals and constraints, use cases, logical views, Interface descriptions, and details on size, performance, and quantity.

2. Architectural Representation

The frontend architecture consists of an HTML webpage which would hold a user-input box where the users can input any Boolean expression, along with a button that would help to submit the expression to the backend program. The interface should be user-friendly, also notifying the user about any error present in their expression, as well as suggesting to them how to fix that error.

The backend consists of a C++ program, following object-oriented programming for this problem statement, i.e., an expression class and following a top-down approach to breakdown the expression and analyzing it before performing any calculations.

3. Architectural Goals and Constraints

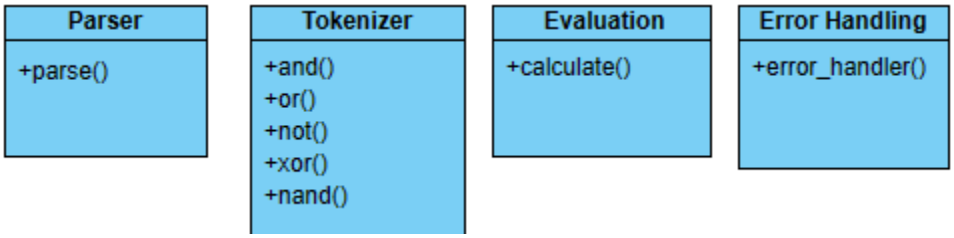
This calculator aims to be efficient, user-friendly, and extensible. We would like to be able to add more features and functionalities in the future. Some constraints include development time, C++ language, and computational efficiency.

4. Logical View

4.1 Overview

The logical view for the Boolean simulator is designed to show all the system's main components and their roles in processing the calculations of the Boolean expressions. This view is crucial for understanding the application of the code to the calculations and how all of it works together.

4.2 Architecturally Significant Design Modules or Packages



5.2.1 Parser

Analyzes and transforms the user-inputted expressions and transforms them into a structured format so that the system can understand it and evaluate the calculations.

5.2.2 Tokenizer

Breaks down all the Boolean expressions into smaller and more understandable components like operators and values for easier processing. It's also responsible for determining the order of operations based on operator precedence and handles parentheses to ensure the correct evaluation of sub-expressions. It also tokenizes the expression into operands, operators, and parentheses.

5.2.3 Evaluation

Evaluates the inputed Boolean expression and whether it correctly encompasses the constraints provided. If yes, then it should go ahead and breakdown the expression into smaller units and evaluate the outcome.

5.2.4 Error Handler

Identifies and handles the errors within the user-inputted expressions, offering feedback to the user to help them correct any syntactical errors or logical mistakes. Catches and properly handles errors that are found in input and parsing process.

5. Interface Description

There will be a Web interface using HTML that allows the user to interact with the Boolean Logic Simulator. The user will be able to input values and operations in a text box and use a button to submit the input into the Boolean Logic Calculator. The Calculator will then display the result.

The input will be a textbox and the output will be displayed on the HTML webpage after the button is pressed to submit the input. If an error occurs, an alert will pop up to notify the user that the input is invalid.

As for the core program, it should consist of a lexical analyzer that would analyze the Boolean expression inputted and return whether it is valid. After confirming its validity, it should move ahead further in the code where it would be broken down into smaller units for calculation. Recursion method could be implemented for this specific task.

6. Quality

The architecture of the Boolean Logic Simulator is created to ensure high-quality, reliability, and user-friendly usability. The system incorporates error handling and a great code design for easy maintenance and future enhancements. User-friendly interfaces are also included in the architecture to prioritize the users' interactions with the software.