RAND Corporated

GUI Boolean Calculator
Software Development Plan
Version 2.0

Revision History

Date	Version	Description	Author
20/02/2024	1.0	Initial Draft at weekly scrum meeting.	Jahnvi Maddila, Shivansh Shrivas, Tej Gumaste, Vamsi Doddapaneni, Jay Patel
25/02/2024	2.0	Final Draft	Jahnvi Maddila, Shivansh Shrivas

Table of Contents

Software Development Plan

-

1. Introduction

Welcome to our comprehensive Software Development Plan, a blueprint designed to guide our team through the successful realization of our semester-long objective: the creation of a sophisticated calculator using the power of C++. This document not only provides a holistic overview of our project but also delineates a systematic plan that ensures the effective distribution of tasks among team members. By delving into our timeline, versioning goals, and collaborative strategies, this plan outlines a strategic approach to foster seamless teamwork, maximize efficiency, and guarantee the delivery of a high-quality software product. Join us on this journey as we articulate a well-structured and organized plan of attack to bring our collective vision to fruition.

1.1 Purpose

The primary objective of this Software Development Plan is to serve as a comprehensive guide for steering and managing the entire project. It is crafted to encapsulate all essential information required for effective project control. This document outlines the strategic approach to software development and stands as the overarching plan that managers utilize to steer the development effort.

Key stakeholders and their uses of the Software Development Plan include:

- **Project Manager**: Utilizes the plan to strategically schedule the project, identify resource requirements, and monitor progress against the established timeline.
- **Project Team Members**: Refer to the plan to gain insights into their specific tasks, timelines, and dependencies. It provides a roadmap for team members to understand what is expected of them and how their activities contribute to the overall project success.

1.2 Scope

This *Software Development Plan* describes the overall plan to be used by the <project name> project, including deployment of the product. The details of the individual iterations will be described in the Iteration Plans. The plans as outlined in this document are based upon the product requirements as defined in the *Vision Document*.

1.3 Definitions, Acronyms, and Abbreviations

• AND Operator:
The AND operator, a logical conjurer, performs the operation of conjunction. It symbolizes the intersection of conditions, producing a true result only when all inputs are true.

• OR Operator:
Behold the OR operator, a master of inclusivity. It symbolizes the logical disjunction, yielding a true result if at least one input is true.

• NOT Operator:
The NOT operator, a logical artisan, excels in negation. It flips the truth value of its input, transforming true into false and vice versa.

• XOR Operator:
Meet the XOR operator, a subtle logic weaver. It stands for exclusive disjunction, producing a true result when the

inputs are distinct (one true, one false).

1.4 References

For the **Software Development Plan**, the list of referenced artifacts includes:

- Iteration Plans
- Vision: To create an efficient Boolean Calculator with a user-centric interface design.
- Any other supporting plans or documentation.

1.5 Overview

This *Software Development Plan* contains the following information:

Project Overview — provides a description of the project's purpose, scope, and objectives. It also defines the deliverables that the project is expected to deliver.

Project Organization — describes the organizational structure of the project team.

Management Process — explains the estimated cost and schedule, defines the major phases and milestones for the project, and describes how the project will be monitored.

Applicable Plans and Guidelines — provide an overview of the software development process, including methods, tools and techniques to be followed.

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

This project constitutes a comprehensive exploration of digital logic, requiring the development of a C++ program functioning as a simplified Boolean logic simulator. The primary goal is to create a robust program capable of simulating the intricate behavior of logic circuits, encompassing essential operations such as AND, OR, NOT, NAND, and XOR. The program's scope extends to the handling of complex logic circuits featuring multiple gates and input/output signals. Beyond the code, we will delve into the foundational concepts of logic gates, truth tables, and expression evaluation, enhancing our skills in parsing, data structures, algorithm design, and software engineering principles.

2.2 Assumptions and Constraints

The Project will be done in C++. We have a team of 5 people, all proficient in Python, but beginners in C++.

2.3 Project Deliverables

Requirements, design specs, test cases, code.
Deliverables for each project phase are identified in the Development Case. Deliverables are delivered towards the end of the iteration, as specified in section *4.1.3 Project Schedule*.

2.4 Evolution of the Software Development Plan

Version 1	rough draft with some of each section completed.

Version 2	is fully completed with each section finalized
Future Version	criteria for future revision. We will change the document if we decide we need to change our process to make it more efficient or faster.

The *Software Development Plan* will be revised prior to the start of each Iteration phase.

3. Project Organization

3.1 Organizational Structure

This project has 5 team members, all of whom will contribute code for the deliverable. We plan on having weekly sprints followed by a sprint review.

3.2 Roles and Responsibilities

Team Lead / Administrator: Jahnvi Maddila

- Contact: jahnvi.maddila@ku.edu, Cell: 913-548-5198
- Responsibilities:
    - overall project leadership and coordination,
    - organizing agendas for each meeting,
    - setting up meetings,
    - final product testing,
    - communicating with the client/product owner,
    - communicating with the team.

Lead Product Engineer: Tej Gumaste

- Contact: tej.gumaste@ku.edu , Cell: 785-424-2555
- Responsibilities:
    - Working on product deliverables,
    - Writing core C++ code,
    - Collaborating with other Developers,
    - Organizing team artifacts.

Quality Assurance Engineer: Shivansh Shrivas

- Contact: shivansh@ku.edu, Cell: 785-423-7806
- Responsibilities:
    - Testing and Reporting Issues,
    - Testing data structures for implementing the calculations,
    - Code Reviews.

Documentation Auditor: Jay Patel

- Contact: jay@ku.edu, Cell: 508-364-7808
- Responsibilities:
    - Taking logs of meetings and updating the repository,
    - Organizing team artifacts and collaborating with other developers,
    - Commenting each section of code for clear readability.

Graphical User Interface (GUI) Engineer: Vamsi Doddapaneni

- Contact: n558d980@ku.edu, Cell: 916-462-0103
- Responsibilities:
    - Creating an intuitive user interface with all desired features,
    - Usability testing,
    - Code writing.

4. Management Process

4.1 Project Plan

This section contains the schedule and resources for the project

4.1.1 Iteration Objectives

- Complete AND
- Complete OR
- Complete NOT
- Complete NAND
- Complete XOR
- Complete GUI

4.1.2 Releases

Release	Functionality
beta 1.0	Add AND, Add OR
beta 1.1	Add NOT, Add NAND
beta 1.2	Add XOR, expression parsing.
demo 1.0	is fully functional in the terminal.
beta 2.0	rough sketch of the graphical user interface.
demo 2.0	first official release
demo 3.0	Final Release (v1.0)

4.1.3 Project Schedule

Date	Task
Febuary 25th	Complete Project Management plan
March 3rd	Complete Project Requirement
March 10th	Project Architecture Requirements
March 17th	Begin Sprints
March 24th	Updating Additional Features
March 31st	Demo 1.0
April 7th	Demo 2.0
April 14th	Project Implementation
April 21st	Test Cases
April 28th	Project User Manual
May 5th	Final Product Release (v1.0)

4.2 Project Monitoring and Control

- Requirements Management: We will study what is needed to have a functioning calculator. We will do artifact elicitation and study how to use C++ and a stack system from reliable resources on how to build a calculator from scratch.
- Quality Control: We will use GitHub to store the code. For quality assurance, we will have team members double-check the code and make sure it runs. We will verify each iteration and version with test input to be sure that proper functionality is achieved through things such as modulus, addition, etc.
- Risk Management: For risk management, we will be sure to check for memory leaks in the C++. We will go through a systematic review where we confirm that the code is working before confirming that an iteration has been properly completed.
- Configuration Management: we will use GitHub to store and control version management. We will use the Agile Method to foster collaboration, adaptability, and iterative development. This approach, combined with GitHub's robust features, ensures seamless coordination among team members, efficient tracking of changes, and the ability to respond quickly to evolving project requirements.

4.3 Quality Control

Defects will be recorded and tracked as Change Requests and defect metrics will be gathered (see Reporting and Measurement below).

All deliverables are required to go through the appropriate review process, as described in the Development Case. The review is required to ensure that each deliverable is of acceptable quality, using guidelines and checklists.

Any defects found during the review that are not corrected prior to releasing for integration must be captured as Change Requests so that they are not forgotten.

4.4 Risk Management

Risks will be identified in the Inception Phase using the steps identified in the RUP for Small Projects activity “Identify and Assess Risks”. Project risk is evaluated at least once per iteration and documented in this table.

4.5 Configuration Management

Appropriate tools will be selected which provide a database of Change Requests and a controlled versioned repository of project artifacts.

All source code, test scripts, and data files are included in baselines. Documentation related to the source code is also included in the baseline, such as design documentation. All customer deliverable artifacts are included in the final baseline of the iteration, including executables.

The Change Requests are reviewed and approved by one member of the project, the Change Control Manager role.

5. Annexes

The project will follow the Agile and UPEDU process.