Whatever Co.

Whatever Calculator
Software Development Plan
Version <1.0>

Revision History

Date	Version	Description	Authors
<19/09/24>	<1.0>	First draft	Michael Buckendahl Ryan Graham Isaac Miller Thad Flournoy Serom Kim Anthony Tran

Table of Contents *[keep this; say N/A when inapplicable]*

Software Development Plan

1. Introduction

The Software Development Plan outlines the main purposes and scope of our arithmetic program. This includes the project overviews, relevant terms with definitions, references, and our organization and processes necessary for the project management, execution, and success of our Software Development project.

1.1 Purpose

The *Software Development Plan* intends to gather all data necessary to manage the project. It is essentially utilized as a guide for the team and serves as the backbone of the creation of our program.

Who will use the *Software Development Plan*:

- The **project manager** will utilize the *Software Development Plan* to maintain deadlines and track progress to ensure a timely submission and completion of the project.
- **Project team members** will utilize the *Software Development Plan* as a reference point to determine what tasks they need to complete and when they need to complete it by.

1.2 Scope

This *Software Development Plan* describes the overall plan to be used by the Whatever Calculator 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

Issues: Can be used interchangeably with defects, goals not met or issues arriving to those goals. Must be documented and reported.

Suggestions: Go in tune with issues, once an issue is documented, suggestions will be made according to the team administrator and assistant team administrator.

Change Requests: Requests for changes to the projects. They need to be documented and approved by team members before making a final commitment.

Deliverables: Outputs that define a specific goal that was created and outlined. Deliverables are to be documented and act as a completed goal.

1.4 References

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

- *Iteration Plans*
- *Development Case*
- *Software Requirements*

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.

Annexes — provides additional relevant information to provide further insight in the requirements and further verification of additional information

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

The purpose of this project is to create an arithmetic expression evaluator in C++ for a larger compiler product for a language L. This evaluator will parse and compute arithmetic expressions while following arithmetic rules for precedence and parentheses. The scope of this project includes building a parser that supports basic arithmetic operations including addition, subtraction, multiplication, division, modulo, and exponentiation. The primary objective of this project is to deliver a functional C++ program that can accurately parse and evaluate arithmetic expressions. Additional objectives include creating detailed documentation for requirements of the program, design speciation's, and a plan to test the program for quality and robustness.

2.2 Assumptions and Constraints

Several assumptions are made for this project. First, the language L's compiler will follow PEMDAS (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) strictly for operator precedence. It is also assumed that only integer arithmetic will be allowed for the initial version of the evaluator. The evaluator will also need to handle edge cases such as division by zero. The constraints for this project include that it must be developed in C++ and follow object-oriented design principles. As well as strict adherence to both the project requirements and deadlines.

2.3 Project Deliverables

- **Requirements Document**: A detailed description of the evaluator's functionality, including supported operators, edge cases, and user interactions. **Due: October 13, 2024**
- **Design Specification**: A comprehensive outline of the system architecture, data structures, algorithms for parsing and evaluation, and the overall approach. **Due: November 10, 2024**
- **Source Code and Implementation:** The Fully implemented C++ code for the evaluator, adhering to the design specifications. **Due: December 12, 2024**
- **Test Cases**: A complete set of test cases to verify the correctness and quality of the evaluator. **Due: December 12, 2024**
- **User Manual**: A user guide that explains how to use the evaluator, with examples of valid and invalid expressions. **Due: December 12, 2024**

2.4 Evolution of the Software Development Plan

Version 1	Initial version with integer arithmetic and basic operator support.
Version 2	Includes support for unary operators and additional error handling.
Version 3	Extends functionally to UI and Edge cases like nested parentheses.

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

3. Project Organization

3.1 Organizational Structure

Our current organizational structure involves six main roles. These roles are the Team Administrator, the Assistant Team Administrator, the Project Leader, the Project Organizer, the Technical Leader, and the Quality Assurance Engineer. It is important to note that each role does not overrule another.

Team Administrator: The team administrator is responsible for making sure that everyone within the team is doing their job effectively. They establish weekly meeting times for the team as well as the goals that each team member should be achieving.

Assistant Team Administrator: The assistant team administrator assists the team administrator in their responsibilities. The assistant team administrator fills in for the team administrator when needed, and double checks the team administrator's actions to ensure that they are reasonable.
Project Leader: The project leader ensures that everything, including documentation and parts of the system, are completed in a timely manner. They also divide up the portions of work on the documentation of the project.

Project Organizer: The project organizer helps assist the project leader in completing all parts of the project in a timely and orderly manner. They also work with the technical leader to help divide up the work that needs to be implemented into the system.

Technical Leader: The technical lead oversees how the group should design parts of the program. When designing the system, the technical lead is in charge of finding the most sustainable and suitable code to implement into the system.

Quality Assurance Engineer: The quality assurance engineer tests the system to find any defects or bugs within the program. If there is an issue with the system, the quality assurance engineer works with the technical lead to find a fix for the problem.

3.2 Roles and Responsibilities

Person	Team Role	Contact Info	Availability	Coding Experience
Serom Kim	Team Admin	serom.kim@ku.edu	M after 3 Tu/Th/Fri after 5 p.m.	Experienced in Python and HTML, Familiar with C and C++.
Anthony Tran	Assistant Team Admin	anthonytran0416@ku.edu	Mo/Wed after 3:00PM Tu/Th after 4:00PM Fri: N/A Sat/Sun Depends on prior notice	Experienced in Python, Familiar with Javascript
Michael Buckendahl	Project Leader	mbuck29@ku.edu	Tu/Th - after 5 Fri – after 2	Experienced in Python, familiar with Java and C
Isaac Miller	Project Organizer	isaac.miller@ku.edu	Tu/Th after 4:00,	Experienced in

			Mo/Wed/Fri after 6:00	Python, Familiar with Java
Ryan Graham	Technical Leader	ryangraham@ku.edu	Mon/Wed after 3:00 PM Tuesday after 4:00 PM and any weekends with prior notice	Experienced with Python, and a modicum of experience with a variety of other languages
Thad Flournoy	Quality Assurance Engineer	t941f863@ku.edu	Mon-Fri after 4:00 p.m.	Experienced with Python and moderate practice with other languages.

4. Management Process

4.1 Project Plan

4.1.1 Iteration Objectives

Each iteration will have the following objectives, which will be ordered as follows:

1. Determine what the program needs to do
    - Supported operators (e.g. +, -, *, /, %, etc.)
    - Order of operations
    - Appropriate response to bad inputs
2. Design the arithmetic interpreter
    - Determine use cases
3. Implement arithmetic interpreter
4. Test arithmetic interpreter
    - Verify valid outputs for valid inputs
    - Ensure interpreter gracefully resolves bad inputs
    - Test any and all edge cases that anyone can come up with
5. Create Documentation
    - Describe all use cases
    - Describe all possible errors
    - Make sure documentation is sufficiently detailed.

4.1.2 Releases

The software is planned to have two releases, an alpha version and a full release; the alpha version is the release that consists of the first functional implementation of the code (akin to version 1 in section 2.4), and the full release refers to a bug-free version that provides a reasonable response to any edge-case (like version 3).

4.1.3 Project Schedule

Iteration Description	**Target Date (Tentative)**
Define Project Requirements	13/10/24
Create Project Design	10/11/24

Code Implementation	12/12/24
Completed Testing	12/12/24
Quality Check, Wrap Up Loose Ends	12/12/24

4.2 Project Monitoring and Control

The requirements for this system are captured in the Vision document. Requested changes to requirements are captured in Change Requests, and are approved as part of the Configuration Management process.

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 review which 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 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.

Refer to the Risk List Document (CCC-DDD-X.Y.doc) for detailed information.

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.

Refer to the Configuration Management Plan (EEE-FFF-X.Y.doc) for detailed information.

5. Annexes

The project will follow the UPEDU process.

Other applicable process plans are listed in the references section, including Programming Guidelines.