348 Group

Algebraic Equation Calculator
Software Development Plan

Version 1.0

Revision History

Date	Version	Description	Author
26/09/24	0.1	Wrote out details for 4.4,4.5,4.7 and 4.8	Kaia Burkholder
29/09/24	0.2		Sam Kelemen

Table of Contents

Software Development Plan

1. Introduction

1.1 Purpose

Develop a C++ program to parse and evaluate arithmetic expressions, supporting basic operations, parentheses, and operator precedence (PEMDAS). The purpose of the *Software Development Plan* is to gather all information necessary to facilitate development of the project. It describes our approach to the development of the software and is the top-level plan generated and used by managers to direct the development effort.

The following people use the *Software Development Plan*:

- The **project manager** uses it to plan the project schedule and resource needs, and to track progress against the schedule.
- **Project team members** use it to understand what they need to do, when they need to do it, and what other activities they are dependent upon.

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, and include the program properly handling +, -, *, /, %, and **, processing numeric constants, managing parentheses, and handling errors like division by zero.

1.3 Definitions, Acronyms, and Abbreviations

OOP: Object oriented programming, a computer programming model that organizes software design around data, or objects, rather than functions and logic.
QA: Quality Assurance

1.4 References

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

- *Iteration Plans:* Not yet created
- *Vision:* Create a fully functioning, robust calculator developed entirely within C++. The program must properly handle +, -, *, /, %, and **, process numeric constants, manage parentheses, and handle errors like division by zero.

1.5 Overview

*[This subsection describes what the rest of the **Software Development Plan** contains and explains how the document is organized.]*

This *Software Development Plan* contains the following information:

Project Overview — Will build an arithmetic evaluator in C++, will take in arithmetic expression as input, parse it, and calculate the result according to PEMDAS.

Project Organization — **Project Manager:** Ensures all team members complete their objectives on time. Monitors project progress to stay on schedule.

Scrum Master: Schedules work sessions. Conducts regular check-ins to track development progress.

Technical Lead: Enforces good coding practices across the team. Resolves technical issues and errors.

Quality Assurance Lead: Ensures the program meets high-quality standards. Conducts tests, including using faulty arithmetic, to identify bugs.

UI Designer: Designs the user interface and creates a visual representation of the interface.

Version Control Manager: Tracks project changes and ensures proper role fulfillment by all team members.

Management Process — The estimated cost for this project is $0. The project schedule is currently unknown, but key deadlines include:

* Project plan due on September 29th.

* Software Requirements Specification (SRS) due on October 20th.

The Project Manager will oversee progress to ensure the project stays on track and is submitted on time.

Applicable Plans and Guidelines — **Development Methodology:** The Scrum Master will manage work sessions and ensure timely progress.

Version Control: Use Git (GitHub) for version control, with regular commits and feature branches. The Version Control Manager will oversee all repository changes.

Coding Standards: Apply Object-Oriented Programming (OOP) principles for modular design, ensuring each component (parsing, operator precedence, etc.) is in a separate class. Additionally, proper commenting and documentation will accompany all code, following industry best practices for clarity.

Tools and Technologies: C++ will be the primary language used for development

Error Handling: Implement robust error handling (e.g., division by zero, invalid expressions) with clear user feedback.

User Interface: Develop a simple command-line interface (CLI) for input and output.

Quality Assurance: The QA Lead will conduct testing based on a detailed test plan, covering all edge cases and ensuring requirements are met.

Documentation: Maintain well-s9tructured project documents and a README file for usage instructions and project overview.

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

Purpose:

Develop a C++ program to parse and evaluate arithmetic expressions, supporting basic operations, parentheses, and operator precedence (PEMDAS).

Scope:
The program will handle +, -, *, /, %, and **, process numeric constants, manage parentheses, and handle errors like division by zero.

Objectives:

- *Implement a parser for arithmetic expressions.*
- *Ensure correct handling of operator precedence and parentheses.*
- *Provide a simple command-line interface.*
- *Incorporate robust error handling.*
- *Deliver a well-tested C++ program with supporting documentation.*

2.2 Assumptions and Constraints

Assumptions:

- All team members have basic proficiency in C++.
- The required development tools (e.g., C++ compiler, version control system) are readily available and free to use.
- Team members will be available for the duration of the project without unexpected absences.
- The project scope will remain stable, with no significant changes in requirements.
- All testing will be done in a local development environment, without external dependencies.

Constraints:

- **Time:** The project must meet the following deadlines:
    - Project plan due on September 29th
    - Software Requirements Specification (SRS) due on October 20th
- **Resources:** The project has a budget of $0, limiting the use of any paid tools or services.
- **Staffing:** The project will rely solely on the current team members without external help.
- **Technology:** The project must be developed in C++ and adhere to object-oriented programming principles.
- **Testing:** All test cases must be developed and executed within the given time frame and resource constraints, e*nsuring the evaluator handles a variety of inputs effectively*.

2.3 Project Deliverables

The following artifacts will be created during the project:

Project Plan, Requirements, and Design Documents

- Target Delivery: September 29th
- This will include the project plan, software requirements, and design specifications detailing the structure and functionality of the arithmetic evaluator.

Test Plan and Test Cases

- Target Delivery: Unknown (To Be Determined)

- A comprehensive test plan with corresponding test cases will be developed to ensure the evaluator's functionality and error-handling capabilities.

Functional C++ Program

- Target Delivery: Unknown (To Be Determined)
- The final C++ program capable of parsing and evaluating arithmetic expressions, with support for various operators and parentheses.

User Manual/README File

- Target Delivery: Unknown (To Be Determined)
- Documentation explaining how to use the program, including example inputs and outputs.

2.4 Evolution of the Software Development Plan

Version	Date	Description	Reason
1.0	9/26/2024	Initial draft of the Software Development Plan	Start of project development

3. Project Organization

3.1 Organizational Structure

The project team consists of Kaia Burkholder as the Project Manager overseeing and reporting on the project’s progress to ensure an ironed result by the deadline. Sam Kelemen, the Scrum Master will facilitate the agile process along with Ruben Pino-Martinez the technical lead, managing technical aspects. Christopher Brush is the UX Designer creating the user experience. Riley Anderson leads Quality Assurance for the project’s reliability and Sean Crosby is the Configuration Manager and will manage the version control system.

3.2 External Interfaces

This project should not interact with any external groups.

3.3 Roles and Responsibilities

Person	**Unified Process for EDUcation Role**	**Languages for Project**	**Contact Info**
Riley Anderson	Quality Assurance	C, C++	randerson1@ku.edu
Christopher Brush	UX Designer	C, C++	cbr@ku.edu
Kaia Burkholder	Project Manager	C, C++	kaiaburkholder21@ku.edu
Sam Kelemen	Scrum Master	C, C++	samkelemen@ku.edu
Ruben Pino-Martinez	Technical Lead	C, C++	rpm@ku.edu
Sean Crosby	Configuration Manager	C, C++	sean.crosby2027@ku.edu

Anyone on the project can perform Any Role activities.

4. Management Process

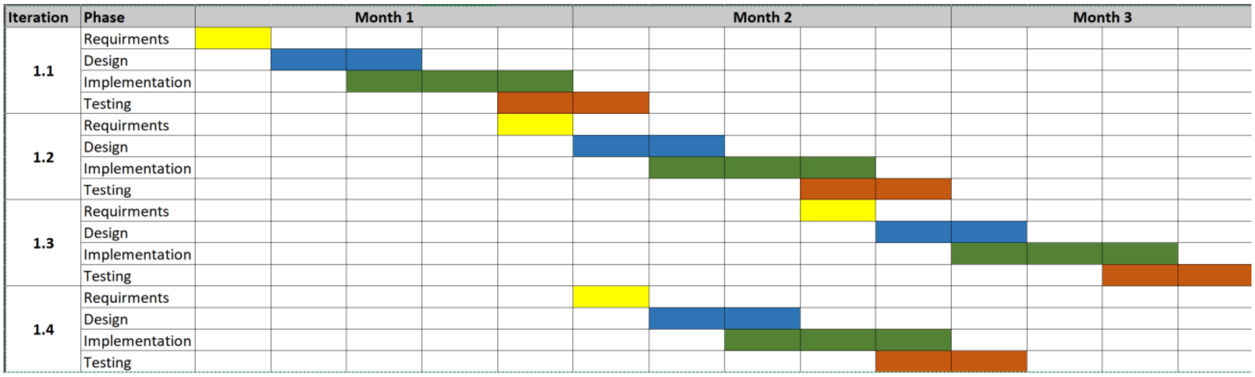
4.1 Project Estimates

There is no cost for this project as everything should already be provided free of charge. The schedule for the project is a meeting weekly with varying levels of time input depending on one’s role in the project. There should not be a re-estimation on cost of the project, however after the third week a re-estimation on time input is possible.

4.2 Project Plan

4.2.1 Phase Plan

Milestone	**Date**
Project Plan: - Designate team rolls, and accurately fill out and submit the Project Plan artifact.	09/29/2024
Requirements: - Thoroughly document the project requirements and fill out the Software Requirements artifact.	10/10/2024
Design: - Design the program structure, and decide the tools to use for development. Create UML class diagrams to document and communicate the program structure.	11/10/2024
Coding: - Implement the models created in the design phase, and make	12/05/2024
Tests	12/12/2024
User's Manual	12/12/2024



Iteration	Phase	Month 1					Month 2					Month 3			
1.1	Requirments	🟨													
	Design		🟦	🟦											
	Implementation			🟩	🟩	🟩									
	Testing					🟧	🟧								
1.2	Requirments					🟨									
	Design						🟦	🟦							
	Implementation							🟩	🟩	🟩					
	Testing									🟧	🟧				
1.3	Requirments									🟨					
	Design										🟦	🟦			
	Implementation											🟩	🟩	🟩	
	Testing													🟧	🟧
1.4	Requirments						🟨								
	Design							🟦	🟦						
	Implementation								🟩	🟩	🟩				
	Testing										🟧	🟧			

Milestones:

1. *Command line parser that can parse expressions with +, -, *, /, %, and ** and integers.*
2. *Add parenthesis to the parser.*
3. *Add error handling to the parser.*
4. *Create a UI for the parser.*

4.2.2 Iteration Objectives

1.1 - Design, implement, and test a command line program that can, given valid input, correctly parse expressions with integers and the following operations: +, -, *, /, %, and **.

1.2 - Design, implement, and test adding the ability to parse parenthesis to the program.

1.3 - Design, implement, and test error handling and add it to the program.

1.4 - Design, implement, and test a user interface for the program.

4.2.3 Releases

1.1 - Demo

A command line program that can, given valid input, correctly parse expressions with integers and the following operations: +, -, *, /, %, and **.

1.2 - Demo

Adds the ability to parse parenthesis.

1.3 - Demo

Adds error handling.

1.4 - Production

Adds a user interface to the program.

4.2.4 Project Schedule

Milestone	**Date**
Project Plan	09/29/2024
Requirements	10/10/2024
Design:	11/10/2024
Coding:	12/05/2024
Tests	12/12/2024
User's Manual	12/12/2024

4.2.5 Project Resourcing

1.1

Training: Team members will need to be proficient in C++ and be able to read and write UML class diagrams. **Target date:**
Staff: All team members will have roles.

1.2
Training: None
Staff: All Team members will have roles.

1.3
Training: None
Staff: All team members will have roles.

1.4

Training: Team members will need to have an understanding of the tools that will be used to create the user interface. **Target date:**
Staff: All team members will have roles.

4.3 Project Monitoring and Control

Requirements Management: The full requirements for the project are found on the project document hosted on the software engineering Canvas page.

Quality Control: Bugs will be fixed throughout the development process as well as in the testing phase, being marked in a change request log found on the GitHub page. Quality Assurance well make sure that all deliverables are working properly

Risk Management: Risks are discussed within team meetings and will be written within the meeting log.

Configuration Management: Git and GitHub will be the primary tools used for version control. The Configuration Manager will ensure that the team works on a single version of the project. Any large problems will likely result in a rollback.

4.4 Requirements Management

The requirements for this program are detailed in the "00-2024-EECS348-Term-Poject" file present on canvas. This is a fixed set of requirements and thus will not need to be managed or updated by the Configuration Manager.

4.5 Quality Control

All bugs found during testing will be recorded in a change request log found on the project GitHub page. Each will be logged with the date found, the error that occurred, and where the break was found.

All deliverables are required to go through the appropriate review process. This review process includes test from the person that worked on the deliverable as well as Quality Assurance to ensure that the deliverable is properly tested.

Any defects found during review which are not corrected immediately must me logged as in the change request log so that it is not forgotten, and other group members can track what needs to be fixed.

4.6 Reporting and Measurement

4.7 Risk Management

Possible risks will be discussed at each stage of the project with team members. This will begin to occur every time the team meets as the implementation stage begins. All potential risks will be logged within the meeting logs as to ensure problems do not go overlooked.

4.8 Configuration Management

Version control will be done through Git and GitHub. All changes will be marked by GitHub and must include a message describing what was changed about a file or what a newly added file used for.

The Configuration Manager will ensure that everyone on the team is using a similar version of the program and that any changes are properly logged.

In cases where the current version breaks due to changes made or pieces of the program are lost, the program will be rolled back to a previous version to ensure that a stable base is maintained.

5. Annexes

The project will follow a version of the Unified Process.

More details on any generic information about the project can be found in the term project document listed under requirements management.