Calculatorz

Arithmetic Expression Evaluator in C++
Software Development Plan
Version 1.1

Revision History

Date	Version	Description	Author
2026-02-03	1.0	Initial draft	Courtney + Group
2026-02-05	1.1	Formatting and polishing	Isaiah
2026 -02-10	1.2	Final touches	Isaiah + Group

Table of Contents

Software Development Plan

1. Introduction

1.1 Purpose

The purpose of the *Software Development Plan* is to gather all information necessary to control the project. It describes the 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 Arithmetic Expression Evaluator in C++ 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 *Software Architecture Document*.

1.3 Definitions, Acronyms, and Abbreviations

See the Glossary in the *User's Manual*.

1.4 References

Project Description, *Software Requirements Specifications*, *Software Architecture Document*, *Test Case*, and *User's Manual*.

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

The goal of this project is to correctly interpret and tokenize given input from the user to parse and evaluate versatile arithmetic operations using C++. The program should be able to interpret the precedence of operations and the proper grouping of input.

2.2 Assumptions and Constraints

We have minimal resources, and every team member is enrolled full-time, with other classes that can't be deprioritized.

2.3 Project Deliverables

Deliverables for each project phase are identified in the Project Plan. Deliverables are delivered towards the end of the iteration, as specified in section 4.2.1 Phase Plan.

3. Project Organization

3.1 Organizational Structure

The Professor provides feedback in the form of grades after each deliverable and is available for contact along with the two teaching assistants. The frontline team is structured as follows. Isaiah as the Administrator, coordinating with Courtney as the Assistant Administrator, facilitates meetings and delegates general tasks. Chase as the Project Leader, coordinating with Taryn as the Assistant Project Leader, defines tasks and delegates specific jobs among the team. Jason as the Technical Leader, defines specific jobs and manages the codebase on the conceptual level. Eian as the Data Administrator and Quality Assurance Engineer, facilitates data transfer of code files and will analyse bug reports.

3.2 External Interfaces

The primary external interfaces for this project include both teaching assistants: Toye Oloko and Liangqin Ren. Their contacts are the following email addresses: toye@ku.edu and liangqinren@ku.edu.

3.3 Roles and Responsibilities

Person	**Unified Process for EDUcation Role**
Team Administrator	Isaiah Jenks
Assistant Team Administrator	Courtney McCray
Project Leader	Chase Hampton
Assistant Project Leader	Taryn Tsosie
Technical Leader	Jason Trieu
Data Administrator/Quality Assurance Engineer	Eian Han

4. Management Process

4.1 Project Estimates

We expect zero financial costs and a 14-week time investment.

4.2 Project Plan

Deliverable one (Project Plan)	February 22
Deliverable two (Software Requirements Specifications)	March 15
Deliverable three (Software Architecture)	April 5
Deliverable four (Implementation)	May 7
Deliverable five (Test Cases)	May 7
Deliverable six (User Manual)	May 7
Project Evaluation	May 7

4.2.1 Phase Plan

Deliverable one will be worked on over two weeks to complete the Project Plan and begin allocating tasks and logging the meeting objectives. Deliverable two will be worked on for two weeks to determine the software requirements needed to implement the project and document the requirements accordingly. Deliverable three will be worked on for three weeks to ensure a methodology for all members to agree on during the implementation phase of the project. All the methodology and break down of the implementation will be logged in a shared document. Deliverable four will be worked on for four weeks to write C++ code to begin creating the project software. Deliverable five will be worked on for two weeks to ensure all bugs and program crashes will be avoided. The code will also be tested to ensure it can handle improper input and redirect the user to enter correct input. Deliverable five will be worked on for one week to create a guide for users to use the project software through clear and concise directions.

4.2.2 Project Schedule & Project Resourcing

Project Phase	Estimated Time Investment	Goal Date for Finish	Actual Deliverable Due Date
Project Planning	2 weeks	Feb. 13	Feb. 22
Software Requirements	2 weeks	Feb. 27	Mar. 15
Software Architecture	3 weeks	Mar. 20	Apr. 5
Implementation	4 weeks	Apr. 17	May 7
Test Cases	2 weeks	May 1	May 7
User Manual	1 weeks	May 7	May 7

4.3 Requirements Management

The requirements for this system are captured in the *Software Requirements Specifications*. Changes to requirements are captured in subsequent versions.

4.4 Quality Control

Defects will be recorded and tracked in *Test Cases*, and defect metrics will be gathered (see Reporting and Measurement below).

All deliverables are required to be submitted for review by the Professor.

4.5 Reporting and Measurement

Updated schedule estimates and metrics summary reports will be generated at the end of each phase in meeting logs.

The Minimal Set of Metrics: Metrics will be gathered on a monthly basis. These include:

Earned time for completed tasks. This is used to re-estimate the schedule and budget for the remainder of the project, and/or to identify the need for scope changes.

Total defects open and closed – shown as a trend graph. This is used to help estimate the effort remaining to correct defects.

4.6 Configuration Management

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

5. Annexes

The project will follow the UPEDU process.

Other applicable process plans are listed in the references section: *Project Description*, *Software Requirements Specifications*, *Software Architecture Document*, *Test Case*, and *User's Manual*.