<Company Name>

<Calculator Project>
Software Development Plan
Version <1.1>

Revision History

Date	Version	Description	Author
9/22/23	1.0	Create a project execution plan.	Chris Harvey, Kristin Boeckmann, Peter W, Sean Brady, Naran Bat, William Morris
12/3/23	1.1	Edit based on changes during implementation.	Peter Walsdorf, Chris Harvey, Kristin Boeckman, Sean Brady, Naran Bat, William Morris

Table of Contents

1. Introduction....4

1.1 Purpose....4

1.2 Scope....4

1.3 Definitions, Acronyms, and Abbreviations....4

1.4 References....4

1.5 Overview....5

2. Project Overview....5

2.1 Project Purpose, Scope, and Objectives....5

2.2 Assumptions and Constraints....5

2.3 Project Deliverables....5

2.4 Evolution of the Software Development Plan....5

3. Project Organization....5

3.1 Organizational Structure....5

3.2 External Interfaces....6

3.3 Roles and Responsibilities....6

4. Management Process....6

4.1 Project Estimates....6

4.2 Project Plan....6

4.3 Project Monitoring and Control....7

4.4 Requirements Management....7

4.5 Quality Control....7

4.6 Reporting and Measurement....7

4.7 Risk Management....8

4.8 Configuration Management....8

5. Annexes....8

Software Development Plan

1. Introduction

This project is to design software which is an application that allows users to interact with a command line interface to input whatever arithmetic they want and receive the correct mathematical expression. The program will be able to add, subtract, multiply, divide, and compute the powers and modulo of any number. It can also handle variable assignment and parentheses operation following the standard operating procedure of PEMDAS.

The results of operation should be validated, expressed, calculated, returned and displayed on the user's screen and be able to be interacted with using the interface. Each character in the input will be examined and determined if it is a valid input. It will check for extraneous and invalid inputs such as extra parentheses, division by zero, and unmatched parentheses/operators.

The program should be able to handle any and all possible errors and edge cases the user could try to/accidentally input while also notifying the user of the error and asking them to change their input/expression.

The goal is to create a useful application which will help the user conduct any mathematical operation they would need for their daily lives.

1.1 Purpose

The purpose of the project is to create a working calculator application. The user should be able to interact with a CLI to input a valid equation, and receive the corresponding numerical output.

1.2 Scope

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

CLI - command line interface
SDP - Software Development Plan
UI - User Interface
cpp - c++

See the Project Glossary.

1.4 References

- No references

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 projects purpose is to create an executable calculator application with a user-friendly interface, encompassing basic arithmetic operations using C++. The project aims to deliver a versatile tool for users to perform a basic range of mathematical operations that one could need in everyday life efficiently.

The expected deliverables include the calculator application itself, testing reports, error handling mechanisms, source code repository, customization options for the user interface, and possibly more. Objectives and deliverables are subject to change with every revision.

2.2 Assumptions and Constraints

Assumptions:

- User Familiarity: Users are assumed to have a basic understanding of calculator functionality and operations.
- Usability: Users are expected to find the calculator’s user-interface intuitive and user-friendly, based on common design principles and user testing.
- Accuracy of Mathematical Operations: The calculator is assumed to perform mathematical operations with high accuracy, adhering to industry standards.
- Stable Development Environment: The development team assumes access to a stable and reliable development environment, including development tools, resources, and support by faculty.
- Team Availability: We assume that all team members will be actively engaged and available throughout the duration of the project, contributing their skills and expertise.
- Prerequisite Knowledge: It is assumed that team members possess a foundational understanding of software development concepts through the class, programming languages, and more, as required by the course prerequisites.
- Access to Learning Resources: We anticipate receiving guidance, clarification, and feedback from our course instructor and TAs to help us succeed in our project.

Constraints:

- Academic Schedule: Our project must align with the college’s academic calendar including adherence to assignment deadlines and the final project submission date specified in the course syllabus.
- Resource Limitation: We acknowledge that the resource limitations may impact our choice of development tools and technologies covered. We will ensure that our project aligns with the scope of the project requirements.
- Project Documentation: All project deliverables, including project plans, design documents, source code, and reports, must align with the documentation standards outlined by the course instructor.

2.3 Project Deliverables

- Iteration 1 deliverables
  - working C++ files that can execute basic arithmetic depending on the file i.e. add.cpp should be able to perform addition, divide.cpp should be able to divide two numbers, etc…
- Iteration 2
  - a working arithmetic class encompassing the files made in the previous iteration.
  - A working user interface
- Iteration 3
  - A universal file that includes all needed functions and a main function that runs everything.

2.4 Evolution of the Software Development Plan

The SDP is a dynamic document that will evolve throughout the project lifecycle. Revisions and updates to the SDP will be conducted based on specific triggers and criteria prior to the start of each Iteration phase to ensure it remains aligned with the project objectives.

3. Project Organization

3.1 Organizational Structure

Sean is our team lead, and he will report to the TAs and Professor while also communicating with our other members about what needs to be done and what our deadlines are. He will keep track of scheduling and taking notes of meetings. Christopher and Kristin are our project leads, and they will help coordinate us through programming. There are two project leads so that we can break up our code into two groups in order to work effectively with smaller parts of the project. The way our project will be viewed is as a calculator application that anyone can use to perform mostly basic operations on.

3.2 External Interfaces

N/A

3.3 Roles and Responsibilities

Person	**Unified Process for EDUcation Role**
Christopher Harvery	Project Lead 1
Sean Brady	Team Lead
Peter Walsdorf	Testing and Reporting
William Norris	Interface Design
Kristin Boeckmann	Project Lead 2
Naran Bat	Debugging and Review

Anyone on the project can perform Any Role activities.

4. Management Process

4.1 Project Estimates

N/A

4.2 Project Plan

4.2.1 Phase Plan

N/A

4.2.2 Iteration Objectives

- Iteration 1

    - Create a main.cpp file which will act as both the run file which launches the user interface as well as contains the used functions.
- Iteration 2
    - Create functions for using operators
    - Create function for parenthesis handling
    - Create function for evaluating expressions
- Iteration 3
    - Combine functions into main.cpp
    - Develop UI
- Iteration 4
    - Create edge case handling and debug and test
- Iteration 5
    - Check for all valid inputs and expressions.
    - Check for all user interaction edge cases and decide how to handle them
- Iteration 6
    - Finalize deliverables and compile the final program for final testing.
    - Release program.

4.2.3 Releases

N/A

4.2.4 Project Schedule

End of Fall 2023 KU Semester

4.2.5 Project Resourcing

N/A

4.3 Project Monitoring and Control

N/A

4.4 Requirements Management

N/A

4.5 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.

4.6 Reporting and Measurement

N/A

4.7 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.

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

4.8 Configuration Management

Appropriate tools will be selected which provide a database of Change Requests and a controlled versioned repository of project artifacts. These artifacts will be managed and tracked inside of a shared GitHub repo.

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.