Doing Math With C++
Software Development Plan
Version 1.0

Revision History

Date	Version	Description	Author
19/09/23	1.0	first attempt to complete project plan completed with team	Wil Johnson, Will Hedges, Eric Loseke
24/09/23	2.0	finalizing document	

Table of Contents

1. Error! Bookmark not defined.

1.1 **Error! Bookmark not defined.**

1.2 **Error! Bookmark not defined.**

1.3 **Error! Bookmark not defined.**

1.4 **Error! Bookmark not defined.**

1.5 **Error! Bookmark not defined.**

2. Error! Bookmark not defined.

2.1 **Error! Bookmark not defined.**

2.2 **Error! Bookmark not defined.**

2.3 **Error! Bookmark not defined.**

2.4 **Error! Bookmark not defined.**

3. Error! Bookmark not defined.

3.1 **Error! Bookmark not defined.**

3.2 **Error! Bookmark not defined.**

4. Error! Bookmark not defined.

4.1 **Error! Bookmark not defined.**

4.2 **Error! Bookmark not defined.**

4.3 8

4.4 9

4.5 9

5. 9

Software Development Plan

●

1. Introduction

This document contains an overview and plan to systematically complete this semester-long goal of creating a calculator using C++. It shows how we will distribute the work amongst group members and what our timeline and version goals shall be. We outline our plan of attack for how our team will most efficiently work together and have high quality. As the book, *Quality is Free* suggests, we can easily attain a professionally done project with just a small amount of planning, good strategy, and hard work.

1.1 Purpose

This *Software Development Plan* is a document that outlines all details of the project, *Doing Math with C++*, before we move on to development. It includes our approach to development of the software, information on our product and scope, and illustrates our project and team structures.

This plan will be used primarily by the following people:

- **The 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 on.

1.2 Scope

This *Software Development Plan* describes the overall plan to be used by *Doing Math with C++*, including deployment of the product. The details of the individual iterations will be described in the Iteration Section.
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

stack: it is a FILO structure that stores values. Typically has a pop(), push(), peek(), and isEmpty() method.

%: This is the Modulus sign, it represents the remainder of division.

/ : this represents division

*: this represents multiplication

^: this represents exponents

1.4 References

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

• Iteration Plans [for example: plan to implement + and -; plan to implement * and /, ...]
• Vision: to create an easy-to-use calculator that can solve complicated expressions

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 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 aim of this project is to create a C++ program that can parse and evaluate arithmetic expressions containing operators +, -, *, /, %, and ^ as well as numeric constants. The program should be able to handle expressions with parentheses to define precedence and grouping. The team will build a versatile arithmetic expression evaluator using C++. The program will take an arithmetic expression as input, parse it, and calculate the result according to the order of operations (PEMDAS).

2.2 Assumptions and Constraints

The Project will be done in C++. We have a team of 5 people, 4 of which learned python last year and have not written much C++ yet. On Tuesdays and Thursdays at 5pm, some group members have weekly Physics Tests. We have almost one semester to complete the project.

2.3 Project Deliverables

Requirements, design specifications, 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 *4.2.4 Project Schedule*.

2.4 Evolution of the Software Development Plan

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

Version 1	rough draft with some of each section completed.	
Version 2	fully completed with each section finalized	
Future Version	criteria for future revision. We will change the document if we decided we need to change our process to make it more efficient or faster.	

3. Project Organization

3.1 Organizational Structure

The project has all 5 team members contributing code for the deliverable. A weekly meeting following the six sigma method, the final product will be user tested by the team lead. Reviews will be done by the entire team to make sure everyone is working together.

3.2 Roles and Responsibilities

Team Lead / Administrator: Manish Singh

- Contact: singhmanish5208@ku.edu, 7854242973
- Availability: Monday 1pm to 5pm, Wednesday 1pm- 5pm, Friday 1pm-3pm
- Responsible for:
- overall project leadership and coordination
- organizing agendas for each meeting
- setting up meetings
- final product testing
- communicating with the client/ product owner

Lead Product Engineer: William Hedges

- Contact: willbhedges@ku.edu, 785-764-5128
- Availability: Monday after 7pm, Tuesday after 5:40pm, Wednesday after 7pm, Thursday after 6pm, Friday 9am-11:50am, 4pm-10pm
- Responsible for:
- working on product deliverables
- reviewing code
- organizing team artifacts and collaborating with other developers

Lead Product Engineer: William Johnson

- Contact: wiljohnson@ku.edu
- Availability: Tuesday/Thursday after 5:00PM, Monday/Wednesday/Friday after 12:00 noon
- Responsible for:
- working on product deliverables
- writing core C++ code
- organizing team artifacts and collaborating with other developers

Product Engineer: Eric Loseke

- Contact: eric.loseke@ku.edu, 424-305-9973
- Availability: Monday 8-11am and after 6:30pm, Tuesday after 5:40-6pm, Wednesday after 6:30pm, Friday 8-11am and after 6:30pm
- Responsible for:
- taking logs of meetings and updating the repository
- organizing team artifacts and collaborating with other developers

Lead Engineer: Achinth Ulagapperoli

- Contact: achinthu@ku.edu
- Availability:every week day after 5 pm
- Responsible for:
- testing data structures for implementing the calculations
- compilation of the program and code reviews
- working on product deliverables
- testing and reporting issues the team ran into to the team lead (or the TA if needed)
- organizing team artifacts and collaborating with other developers

4. Management Process

4.1 Project Plan

This section contains the schedule and resources for the project.

4.1.1 Iteration Objectives

- Complete Addition
- Complete Subtraction
- Complete Multiplication
- Complete Division, Parenthesis, Modulus

4.1.2 Releases

Release	**Functionality**
beta 1.0	Add Addition
beta 2.0	Add Substraction
beta 3.0	Add Multiplication
beta 4.0	Add Division
beta 5.0	Add Parenthesis
beta 6.0	Add Modulus function
6.1	First Official Release
6.2	Tested Official Release

4.1.3 Project Schedule

Date	**Task**
September 19	Complete Project Management Plan
September 26	Write Project Requirements
October 3	Project Architecture Requirements
October 10	Write Code for Addition and Substraction

October 17	Write Code for Multiplication and Division
October 24	Write Code for Parenthesis, Module
October 31	Project Implementation
November 14	Test Cases
November 28	Project User Manual
December 5	Project Due

4.2 Project Monitoring and Control

For 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.

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.

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.

For configuration management we will use git hub to store and control version management. We will use the waterfall method where we will be sure to get each section done before moving on. For example, we will be sure to fully complete the requirements phase and then have all group members verify that each section has been thoroughly checked. For code review, we will use multiple test code cases to ensure that the code functions. We will name each document by the phase it's in like Requirements and what version of it it is. Like Design 1.0. We will share these documents on google docs and we will collaborate on each document together.

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. This is a school project so the only risk is getting a poor grade. There is no financial risk or physical risk that could come. There could be risks in C++ controlling the memory and having that crash and so we will be diligent to have our program properly handle the memory so the program doesn't crash.

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 UPEDU process.

Other applicable process plans are listed in the references section.

We will use the Waterfall Method where we get all the requirements, then design, etc.