Group 22

Arithmetic Parser
Software Development Plan
Version <1.0>

Revision History

Date	Version	Description	Author
26/09/24	1.0	First iteration of Project Plan Outline	

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

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 6

3.1 Organizational Structure 6

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 8

4.5 Quality Control 8

4.6 Reporting and Measurement 8

4.7 Risk Management 8

4.8 Configuration Management 9

5. Annexes 9

Software Development Plan

1. Introduction

This introduction to this arithmetic parser will entail details on the purpose, scope, definitions, acronyms, abbreviations, references, and overview of our software development project.

1.1 Purpose

The overall purpose of our software development plan is to visualize what our system is going to look like as an end result, designate rolls and responsibilities to each member more clearly, and observe if we've kept track of our original plan as changes become implemented over time.

1.2 Scope

The Software Development Plan covers the entire development life cycle of the Arithmetic Parse, including:

- Requirement gathering and analysis
- Design and architecture plan
- Implementation of the Arithmetic Parser features
- Testing and validation
- Deployment and user documentation

1.3 Definitions, Acronyms, and Abbreviations

Deliverables – Project artifacts that will be submitted on or before each due date.

Arithmetic Parser – Program that will take input and be able to parse said input including: +, -, *, /, %, **, and parenthesis.

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Software Development Plan**. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.*

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

- *Iteration Plans*
- *Development Case*
- *Vision: Create an arithmetic parser that given an mathematical expression as an input, evaluates operations add (+), subtract (-), multiplication (*), division (/), modulo (%) and exponentiation (**) and handle parenthesis. It should handle errors gracefully and guide the user as to how to properly use the tool.*
- *Glossary*
- *Any other supporting plans or documentation.*

1.5 Overview

This *Software Development Plan* contains the following information:

Project Overview — Main goal is to create a calculator/arithmetic parser that takes into account other expressions of mathematic operators.

Project Organization — One individual will manage the team, one will develop the interface for the project, one will do the iterative coding, and the other 3 members will do additional debugging/testing

Management Process — Cost deficit in time: Approximately 20-30 hours to create the final product, Average Meeting Periods: 1-2 Times a Month, Method to Manage Progress: Check-ins with other groups members and checking the progress logs

Applicable Plans and Guidelines — Goal to utilize C/C++ to create a rough skeleton for the project and over time iterate throughout the code to make improvements until the project is polished.

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

This project's objective is to create a quality arithmetic parser that can solve various mathematic equations quickly and properly. The parser will be able to perform computations commonly used in engineering making it a handy tool for college students. The main deliverable of this project is an intelligent, quick, and easy to use calculator tool that will make solving complex equations easier for the masses,

2.2 Assumptions and Constraints

- Assumptions: Completion of EECS 268 (Programming 2)
- Constraints: Program language (C++), Equipment (Personal Computers & Cycle Servers), Schedule (Availability here), due dates (listed below)

2.3 Project Deliverables

- Project Plan – 09/29/24
- Project Requirements – 10/20/24
- Project Architecture and Design – 11/10/24
- Project Implementation – 12/12/24
- Project Test Cases – 12/12/24
- Project User Manual – 12/12/24

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.

3. Project Organization

3.1 Organizational Structure

This project will be completed using the roles below in 3.3, but we will be flexible with these roles and work as a team to fill in gaps in each other's skillsets.

3.2 External Interfaces

3.3 Roles and Responsibilities *[the more details here, the easier your job; include contact info, availability info, expertise, …]*

Person	Unified Process for Education Role	Programming Language Knowledge, Computing Platform Experience
Janna Dungao: jannadungao@ku.edu	Product Owner	C, Python, VS Code
Yoseph Ephrem: yephrem@ku.edu	Scrum Master	Python, C, Command Line pro, VS Code
Beckett Malinowski: beckmalinowski@ku.edu	UI/UX Designer	C/C++, Terminal/Command Line, SQL
Shane Huang: shane.huang@ku.edu	Developer/Tester	C, HTML, CSS, SQL, Java, Javascript, Python, VS Code
Dalen Journigan: dalenrashad13@ku.edu	Developer/Tester	Python, VS Code
Mahdi Essawi: Mahdi.essawi@ku.edu	Developer/Tester	Python, JS, VS Code

Availability Link

Anyone on the project can perform Any Role activities.

4. Management Process

4.1 Project Estimates

4.2 Project Plan

Artifact	Due Date
Project Management	09/29/24
Project Requirements	10/20/24
Project Architecture and Design	11/10/24
Project Implementation	12/12/24
Project Test Cases	12/12/24
Project User Manual	12/12/24

4.2.1 Phase Plan

[Include the following:

- *a Gantt chart showing the allocation of time to the project phases (Not necessarily detailed to the activity level; this type of Gantt Chart is providing along with the Iteration Plans themselves; Provide an Overview of the project Timeline with the major miles stones]*
- *identify **major milestones** with their achievement criteria*

Define any important release points and demos.]

*[If available, refer to the related **Iteration Plan Documents** for more details]*

4.2.2 Iteration Objectives

- Project Management Plan: Complete project plan document that overviews roles and requirements of the project.
- Project Requirements: Discuss and finalize what use cases and other requirements to implement into the project.
- Project Architecture and Design: Create a blueprint like document outlining what the software will look like.
- Project Implementation: Coding the project in C++.
- Project Test Cases: Create possible test cases for error-checking code.
- Project User Manual: Complete the document that aides' users in the usage of our completed project.

4.2.3 Releases

We will have multiple software releases throughout the development process

- *Alpha release: Internal testing for the core functionalities of the project*
- *Demo release: Expanding access to few individuals for review and testing purposes*
- *Final launch: Fully tested to no errors, release and submission of the product.*

4.2.4 Project Schedule

**All dates subject to change*

Phases	*Design*	*Implementation*	*Testing*	*Alpha Release*	*Testing*	*Demo Release*	*Final Testing*	*Final Launch*
1	*9/26*	*10/15*	*11/01*	*N/A*	*11/1*	*N/A*	*11/20*	*N/A*
2	*10/03*	*10/20*	*11/05*	*N/A*	*11/13*	*N/A*	*11/22*	*N/A*
3	*10/10*	*10/26*	*11/07*	*N/A*	*11/15*	*N/A*	*11/24*	*N/A*
4	*10/13*	*10/30*	*11/10*	*11/10*	*11/17*	*11/17*	*11/26*	*Deadline day*

4.2.5 Project Resourcing

4.3 Project Monitoring and Control

- Requirements Management: Code will be tested thoroughly during construction and comprehensive unit/integration tests will be written to confirm the accuracy of the final product. All unit tests must be

passed before deploying the final product.

- Quality Control: Quality assurance will involve regular testing, adherence to coding standards, and reviews of project artifacts.
- Risk Management: Risks will be identified during project planning and monitored throughout development, with mitigation strategies documented.
- Configuration Management: Problems and changes will be discussed in our project team's messaging group. Majority agreement on a topic will result in a change to the project. Each non-program-related file will follow the naming scheme <0n-Document-Name>. Any iterations will be considered Version <n.n>. The file and naming system for programming files will be precise and clear as to what they are achieving. The final executable will be under the name arithmetic_parser.exe. All files will be compiled via a compile management tool such as **makefile**. Version control will be managed using git to track changes in project artifacts.

4.4 Requirements Management

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.

4.5 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

Updated schedule estimates, and metrics summary reports, will be generated at the end of each iteration.

The Minimal Set of Metrics, as described in the RUP Guidelines: Metrics will be gathered on a weekly basis. These include:

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

Acceptance test cases passing – shown as a trend graph. This is used to demonstrate progress to stakeholders.

Refer to the Project Measurements Document (AAA-BBB-X.Y.doc) for detailed information.

4.7 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.

4.8 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.