EECS 348 Group 10

Web Calculator for University of Kansas Students
Software Requirements Specifications

Version 1.0

Revision History

Date	Version	Description	Author
10/12/2023	1.0	Initial version	Harlan Williams

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 4

2. Overall Description 4
   - 2.1 Product perspective 4
     - 2.1.1 System Interfaces 4
     - 2.1.2 User Interfaces 4
     - 2.1.3 Hardware Interfaces 4
     - 2.1.4 Communication Interfaces 5
     - 2.1.5 Memory Constraints 5
     - 2.1.6 Operations 5
   - 2.2 Product functions 5
   - 2.3 User characteristics 5
   - 2.4 Constraints 5
   - 2.5 Assumptions and dependencies 5

3. Specific Requirements 5
   - 3.1 Functionality 5
     - 3.1.1 Allow users to input expressions, have them be evaluated, and display results of evaluation 5
     - 3.1.2 Allow users to create and evaluate scripts 6
     - 3.1.3 Allow users to use scripts created and shared by other users 6
     - 3.1.4 Account creation so that users can create, save, and share their own scripts 6
     - 3.1.5 Allow users to login to their accounts 6
   - 3.2 Use-Case Specifications 6
   - 3.3 Supplementary Requirements 6
     - 3.3.1 Handle Errors and invalid expressions 6
     - 3.3.2 Securely store user passwords 6
     - 3.3.3 Store a history of expressions input by the user 6
     - 3.3.4 Project finished by 12/5/2023 6
     - 3.3.5 Project to be implemented in C++ 6
     - 3.3.6 Project includes a README 6
     - 3.3.7 Project must be open-source 7
     - 3.3.8 Project must be able to be hosted as a droplet on DigitalOcean 7

4. Classification of Functional Requirements 7

Software Requirements Specifications

1. Introduction

1.1 Purpose

This document describes the functional and non-functional requirements of the calculator project, as well as the constraints under which the project resides. This document intends to describe all functionality provided by the project that will be visible to the end-user and will be referenced during the design stage of the project in order to guide the development team's decisions.

1.2 Scope

This document describes requirements for both the calculator subsystem and the web-application subsystem. It contains a description of functional requirements important to the core functionality of the application, non-functional requirements which improve the usability of the application for the end-user, and constraints which are set forward by both the project requirements of EECS348 and the particular need for the project to run on Linux and be hosted as a web application on DigitalOcean.

1.3 Definitions, Acronyms, and Abbreviations

This document uses the same definitions, acronyms, and abbreviations described in the Project Plan.

1.4 References

This document references the Project Plan for definitions, acronyms, and abbreviations, as well as the use-case diagram at use-cases.pdf in the project documentation folder.

1.5 Overview

The document first presents a high-level description of all external factors that affect the project from a software and hardware view, the general requirements of users and the type of user that is expected to use the project, and what factors act as constraints on the project's design. Then, a list of specific requirements is presented, along with a brief description of the requirement. These are both functional and non-functional requirements. The constraints are then also enumerated along with a brief description of where the constraint arises from.

Finally, the functional requirements are classified as either *Necessary*, *Desirable*, or *Optional.* Necessary functional requirements are non-negotiable and must be fulfilled for the project to function at all as a calculator. Desirable requirements are those which are secondary to the project's primary goal to implement a calculator. Optional requirements extend the basic functionality of the project but can be removed from the project as well without affecting core functionality.

2. Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

The project must run in a Linux virtual machine environment, and as such must use POSIX system calls whenever accessing OS operations.

2.1.2 User Interfaces

The primary user interface will be through a web app that will provide web pages for evaluating expressions, creating scripts, viewing history, and managing user-accounts.

2.1.3 Hardware Interfaces

The project does not itself manage hardware interfaces such as sockets for communication.

2.1.4 Communication Interfaces

The web application will use the cpp-httplib library to handle outbound and inbound web connections. At a low-level, this will involve the use of OS sockets and will necessitate port forwarding to be hosted on the web.

2.1.5 Memory Constraints

The web application is constrained to use less than 512 MiB (mebibyte) of memory to be able to be hosted on a basic-tier DigitalOcean droplet.

2.1.6 Operations

The application will take user requests through an HTTP interface and respond in a similar way. On the back-end, the project will process user input, either evaluating an expression with the calculator that the project will also develop, or handle database operations such as storing scripts, saving and returning history, or managing user accounts.

2.2 Product Functions

The application will provide an environment in which users can evaluate arithmetic expressions. To do this, it provides a calculator interface, and as well as history and scripting functionality so that users can save previous calculations and make them more general.

2.3 User Characteristics

The application is being developed with college students in mind, but the scripting functionality will allow many kinds of users to extend the application to fit their needs by the creation and sharing of scripts.

2.4 Constraints

The project must be implemented in C++ and must be able to compile and run on a DigitalOcean Linux virtual machine environment. To achieve this, the project must use POSIX system calls whenever interacting with the operating system. The project must also be released under an open-source license to permit graders access to the source code, as well as to permit the project to be extended in the future.

The project is being developed during the Fall 2023 semester of EECS348 Software Engineering I, and thus must be in a completed and gradable state by the end of the semester: December 5th, 2023. Because the project is being developed for a class, it must contain a README and various other documentation about the development process to ease grading as well as to allow graders, and potential future users, to easily learn how to use the application or to contribute to the project under open-source.

2.5 Assumptions and Dependencies

The project is dependent upon the following open-source libraries: Nlohmann JSON for Modern C++, cpp-httplib, Inja, and sqlite3. The project also will depend on DigitalOcean for hosting the web app and assumes that DigitalOcean will be able to meet that need.

3. Specific Requirements

3.1 Functionality

3.1.1 Allow users to input expressions, have them be evaluated, and display results of evaluation

The user will have access to a web interface where they can input an arithmetic expression as text, then submit the expression to the application. The application will then either calculate the result and return it to the user, or indicate if an error occurred, either due to the user inputting incorrectly formatted arithmetic, or a server-side error. Arithmetic expressions consist of positive and negative numbers, addition, subtraction, multiplication, division, modulo, and exponent operators, and parentheses. The expressions will be evaluated according to the PEMDAS order of operations.

A history of expressions will be stored for the duration of the session if the user is not logged in, or permanently if the user has an account and is logged in. The session history will be displayed on the same page that input is requested on, and from the account page, the user will be able to see their history across all sessions.

3.1.2 Allow users to create and evaluate scripts

The user will also have access to a page that allows them to create *scripts*: a series of arithmetic statements along with variables that the user can set when evaluating the script. The application will provide error checking when the user finalizes the script to check for malformed input. When the user accesses a script, it will be provided on a separate page of the web application where the text of the script is immutable, and the user will be prompted for values of the variables found in the script. Once inputted, the user can evaluate the statements with the variables provided, and receive an answer, or an error if the user input was invalid.

3.1.3 Allow users to use scripts created and shared by other users

Scripts can be shared by providing permanent URLs to the page with the finalized script and variable input discussed in 3.1.2. Scripts will have an option to be edited by anyone with an account, and any changes the user makes will create a new script with a new unique URL.

3.1.4 Account creation so that users can create, save, and share their own scripts

In order to store persistent data for users, the user will have to create an account to tie that data to. The account creation process will only involve choosing a username and password to keep account creation simple. The password will at minimum be hashed in the database, but the user should be prompted not to use a password they commonly use as security cannot be guaranteed.

3.1.5 Allow users to login to their accounts

The user will be able to log into their account after account creation and access their user page, which shows them their history, an interface to write scripts in, and a history of previously used or saved scripts.

3.2 Use-Case Specifications

See the use-case diagram 'use-cases.pdf' in the project documentation folder.

3.3 Supplementary Requirements

3.3.1 Handle errors and invalid expressions

The calculator should be able to inform users when they enter an invalid expression and possibly help the user correct it.

3.3.2 Securely store user passwords

Users will have to create an account and set a password to access history and save scripts because these features require persistence.

3.3.3 Store a history of expressions input by the user

A history function to show the user's previous calculations will help users remember their work and reference their past calculations for errors.

3.3.4 Project finished by 12/5/2023

Requirement set by EECS348.

3.3.5 Project to be implemented in C++

Requirement set by EECS348.

3.3.6 Project includes a README

Allows users to quickly learn how the project is set up and how to use it.

3.3.7 Project must be open-source

Allows other contributors to extend the project after it is originally completed for EECS348.

3.3.8 Project must be able to be hosted as a droplet on DigitalOcean

Required if the application is to be available online.

4. Classification of Functional Requirements

Functionality	**Type**
Perform: Addition, Subtraction, Multiplication, Division, Modulo, and Exponentiation	Essential
Perform operations in PEMDAS order of precedence	Essential
Prompt user for input and return the evaluated expression, and reject invalid expressions	Essential
Allow users to create, use, and share scripts with variable assignment	Desirable
Project must be able to be hosted on a DigitalOcean droplet as a web application	Desirable
Handle errors/invalid expressions by returning the error to the user	Optional
Store history of expressions evaluated by the user	Optional
Project must be open-source	Optional