CIS-11 Project Documentation Template

Team Name
The Widdle Wabbits
Team Members
Luis Lopez
Luke Moua
Ivan Hernandez
Project Name
5/25/25

Advisor: Kasey Nguyen, PhD

Part I – Application Overview

This part of the requirements document serves to present the "big picture" of the application. Here you lay out the objectives of the application, how it fits into the business process of the company, and how it relates to other software systems. The sections listed below should be included in this part of the requirements document.

Objectives

In this section you state the commonly accepted objectives of the project.

You must determine the business objectives of the project early on; without clear objectives your project has little chance of succeeding anyway so it does not make sense to move on until the objectives are agreed upon.

Why are we doing this?

To elicit the objectives, ask the business expert, the development manager, and the project sponsor the following questions:

- **What business objectives of the company will this project help achieve? Possible objectives might be reducing costs, improving the customer service, simplifying the workflow, replacing obsolete technology, piloting a new technology, and many others. Also, make sure you understand exactly how the proposed project will help accomplish the stated objective.**

  The purpose of this project is to simplify workflow. Making sure that professors are more efficient at grading and are able to focus more on the students than having to keep up with grading.

- **Why are we doing this project now? What will happen if we do it later? What if we do not do it at all?**

  Doing this project now because it has come to our attention that some professors are having issues with keeping up grading. Pushing this project back further will cause professors to have a backlog of assignments to grade.

- **Who will benefit from this project? Do the people who will benefit from it consider it the most important improvement that can possibly be made at this time? Should we be doing a different project instead?**

  The people that will benefit from this project will be the professors and students. Professors are able to focus more on the students because there is less to grade while students are able to interact with professors more often.

Business Process

The existing process for grading is currently a tedious one. Professors must manually input grades and calculate the average score to share with the class. Having to deal with this tedious task prevents professors from better preparing for lecture material and interrupts their sleep schedule. In addition, it builds up anxiety for students who are worrying for their grades to get posted even though it is not the professor's fault. It is mentally exhausting for one person to grade hundreds of assignments per week. That responsibility can be too much to bear which can negatively affect a professor's mental health. The true criminal to this dilemma is not in the professor or the student but the grading system itself.

Our new grading system introduces a solution to a problem that will benefit both professors and students alike. It will allow grading to be sorted and easily organized into a specific grade. All while also calculating the average score from the inputted test scores. This improved system will reduce stress levels of professors and ease anxiety from students anticipating their grade scores. Our new grading system will implement a feature where exam scores can be inputted into the system, individually assign a letter grade based on the score, and simultaneously calculate the average score. It will do more while inputting the same same amount.

User Roles and Responsibilities

In this section you describe who the users are and how the system fits into what they do.

User	*Responsibilities*	*Business Process*	*Objectives*
Professors	*They are tasked with consistently grading assignments daily and exams every couplee weeks*	*Our new system will make their jobs more time manageable and stress-free*	*Meet time restraints, Supply good feedback to students' exams, Multitask grading for multiple classes*

Production Rollout Considerations

In this section you describe the strategy for production rollout.

In addition, either this section, or an appendix in the requirements document, or a separate document should include the discussion of populating the system data for rollout and the discussion of the expected data and transaction volume.

After making sure the project is tested and working, It will be presented to education companies like college board for it to be distributed. It will be a very easy download from our website which can be accessed if you have internet. The download will install an app and when opened will have all the working code.

Terminology

Our new system is designed to improve on time management for professors and ease the anxiety for students.

Our implementation and improved system are designed to create a simpler grading system that allows its users to input the same amount of data but have more output.

Part II – Functional Requirements

This part of the requirements document states in a detailed and precise manner what the application will do.

The purpose and functionality of this program is to help users with the tasks of making calculations with simple user input interaction. Our goal is to utilize LC-3 and create a user friendly interface for users attempting to calculate test scores. Users will be able to enter five two digit numbers into our program. Where calculations will be performed to display the average , minimum, and maximum scores. Each of these scores will also display a respective letter grade for our users.

Statement of Functionality

In this section you state **precisely** *what the application will do.*

This part, more than anything else in the requirements document, spells out your contract with your customers. The application will ***include all functions listed here and will not include any of the functions not listed****.*

- *Easy to use user interface*
- *Provide Minimum test score*
- *Provide Maximum test score*
- *Provide Average test score*
- *Provide the respective and correct letter grade for each score*
- *The program will be able to properly store user input inside the program's memory*

In this section you must use as precise language as you can since the developers will use it to code the application. When reviewing this part with other people you should pay extreme attention to removing any possibility for ambiguous interpretation of any of the requirements.

Programmers/developers will program a project in LC-3 assembly that will not only provide a great and user friendly experience but also one with proper functions and required output. It is required for the developer to provide appropriate addresses for input and output. Display all requirements including: minimum test score, maximum test score, average test score, and letter grade. The programmer will implement subroutines that make the correct calculations to achieve the goals of the user. The maximum score subroutine display and store the larger score and store its value in a respective register. The minimum score subroutine will display and store the smallest value into a respective integer. The average score subroutine will add all values and divide then by five, stored result in a respective register. Each of the scores will be passed through a algorithm that assigns a letter grade to each of these stored values (e.g. A, B, C, D, F). The programmer will successfully display all functionality to the user. The programmer will also utilize the stack and correctly use a pointer within the program.

If your application has several distinct categories of users, you can list the requirements by user category. User categories may be defined in terms of their job title (clerk, manager, administrator), the frequency with which they will use the system (heavy or casual), the purpose for which they will use the system (operational decisions, long-term decisions), etc. If each category of users uses the system in its own way, structuring the requirements based on user category will make sense.

Our goal is for our to target those who are either working at a school or those who are interested in making calculations through our program.

The teacher will be a heavy user whereas a student would be a casual user. Our goal is to make a simple and easy to use program for all users.

Teacher (Heavy User)

- *Role: Educator*
- *Purpose: Analyze Student performance*
- *Needs: Fast and efficient, easy to use.*
- *Input: Will only enter test scores*

Student (Casual User)

- *Role: Individual*
- *Purpose: personal insight*
- *Needs: Simple interface, not difficult to use*
- *Input: Will only enter test scores*

If your application deals with several kinds of real-world objects, you can list the requirements by object. For example, for a reservation system a booking is an important object, and you may want to list all requirements pertaining to bookings in one sub-section.

Requirements:

- *Valid test scores ranging from : 0 – 100*

Letter Grade Assignment Criteria:

- *A : 90 – 100*
- *B: 80 – 89*
- *C: 70 – 79*
- *D: 60 – 69*
- *F: 0 –59*

These requirements allign with the real world standard grading system.

One of the most common approaches is to list the requirements by feature. For example, features of a word processing application are file management, formatting, editing, etc.

Requirements by Feature:

- *Easy to use user interface*
- *Provide Minimum test score*
- *Provide Maximum test score*
- *Provide Average test score*
- *Provide the respective and correct letter grade for each score*
- *The program will be able to properly store user input inside the program's memory(properly utilize the stack)*
- *Valid test scores ranging from : 0 – 100 and a letter grade assignemnt*
- *Letter Grade Assignment Criteria:*
- *A : 90 – 100*
- *B: 80 – 89*
- *C: 70 – 79*
- *D: 60 – 69*
- *F: 0 –59*

Scope

In this section you state what functionality will be delivered and in which phase.

You should include this section if your development consists of multiple phases. As an alternative to this section, you can note the planned project phase for each feature in the functionality statement section. Usually, it is better to include a separate scope section for easy reference and communication.

- *Phase 1 User interface/inputs:*

  The first thing presented to the user is going to be the interface asking to input 5 test scores. The interface will provide instructions to use and show the outputs after 5 test scores have been inputed

- *Phase 2 processing:*

  This phase will do all the hard work. Taking in the inputs from phase 1 and finding the maximum, minimum and average of all the scores. It will then put those numbers into a converter that transforms it into letter grades.

- *Phase 3 output:*

  after phase 2 the outputs will be shown on the interface completing the project.

Performance

In this section you describe any specific performance requirements.

You should be very specific and use numeric measures of performance. Stating that the application should open files quickly is not a performance requirement since it is ambiguous and cannot be verified. Stating that opening a file should take less than 3 seconds for 90% of the files and less than 10 seconds for every file is a requirement.

Instead of providing a special section on performance requirements, you may include the relevant information for each feature in the statement of functionality.

The performance of our program is greatly emphasized with the requirements of each of our features. Our program must require a simple and easy to use interface that performs well. We take into consideration how we utilize code to display the best results for our users. Our program will accurately display all calculations that are required. These requirements include the minimum, average, and maximum test scores. We hope to ensure a program that has a quick and competitive runtime.

Usability

In this section you describe any specific usability requirements.

You need to include this section only if there are any "overarching" usability goals and considerations. For example, the speed of navigation of the UI may be such a goal. As in the previous section, use numeric measures of usability whenever possible.

A goal for this project is an easy-to-use interface. To accomplish this the program will first ask a simple prompt like "Input 5 test grades and after each one press enter." Then after 5 seconds it will show the max, min, and average in a neat easy to read format Like

example

Max: 90

Min: 40

Avg: 66

Documenting Requests for Enhancements

There does come a time when the requirements for the initial release of your application are frozen. Usually, it happens after the system acceptance test which is the last chance for the users to lobby for some changes to be introduced in the upcoming release.

Currently, you need to begin maintaining the list of requested enhancements. Below is a template for tracking requests for enhancements.

Date	Enhancement	Requested by	Notes	Priority	Release No/ Status
5/23/2025	Add error checking	Luis Lopez	Add error checking- no negative vales/ or values larger than 100	High	Pending

			should be accepted by the program.		

Part III – Appendices

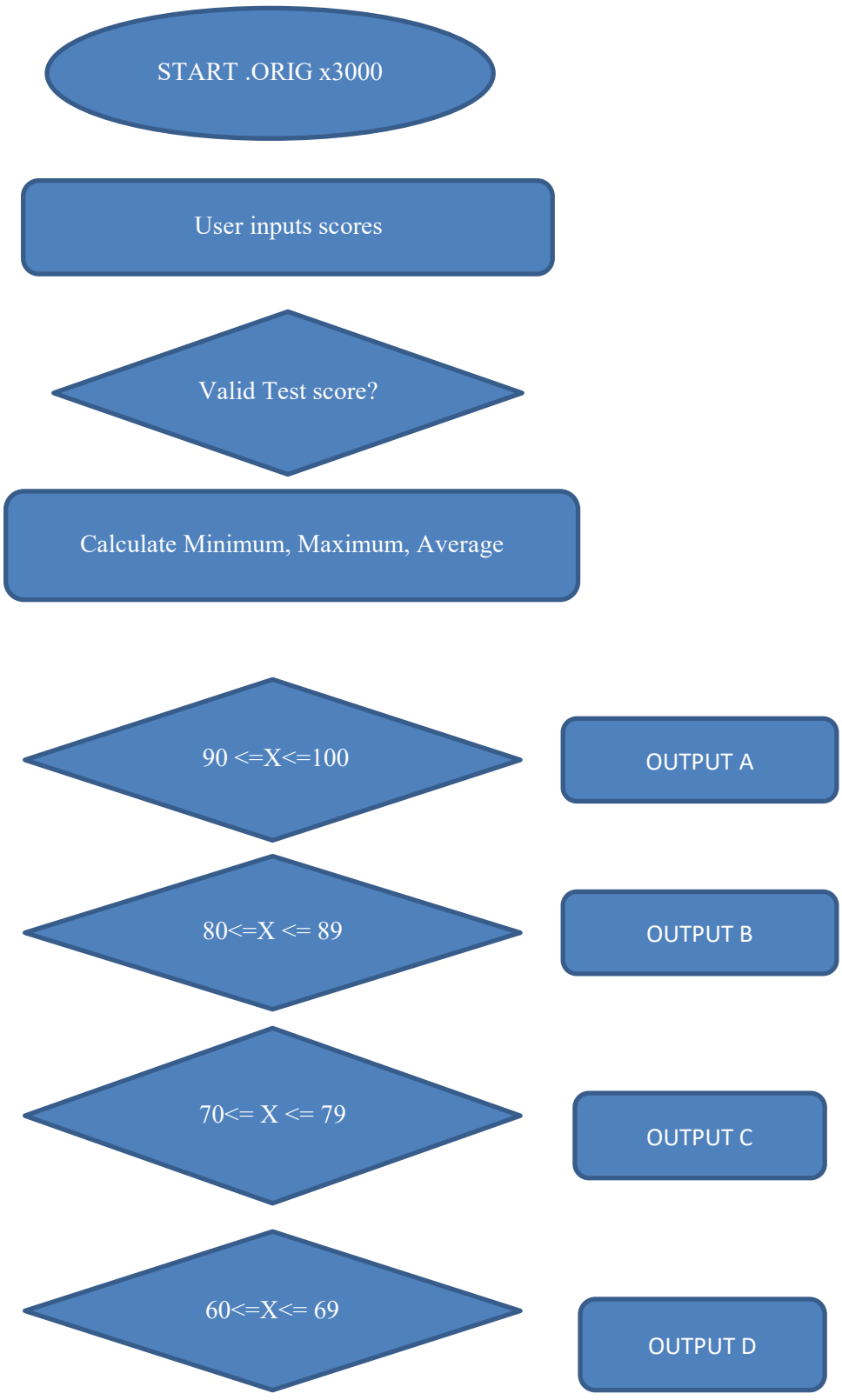
Appendices are used to capture any information that does not fit naturally anywhere else in the requirements document yet is important. Here are some examples of appendices.

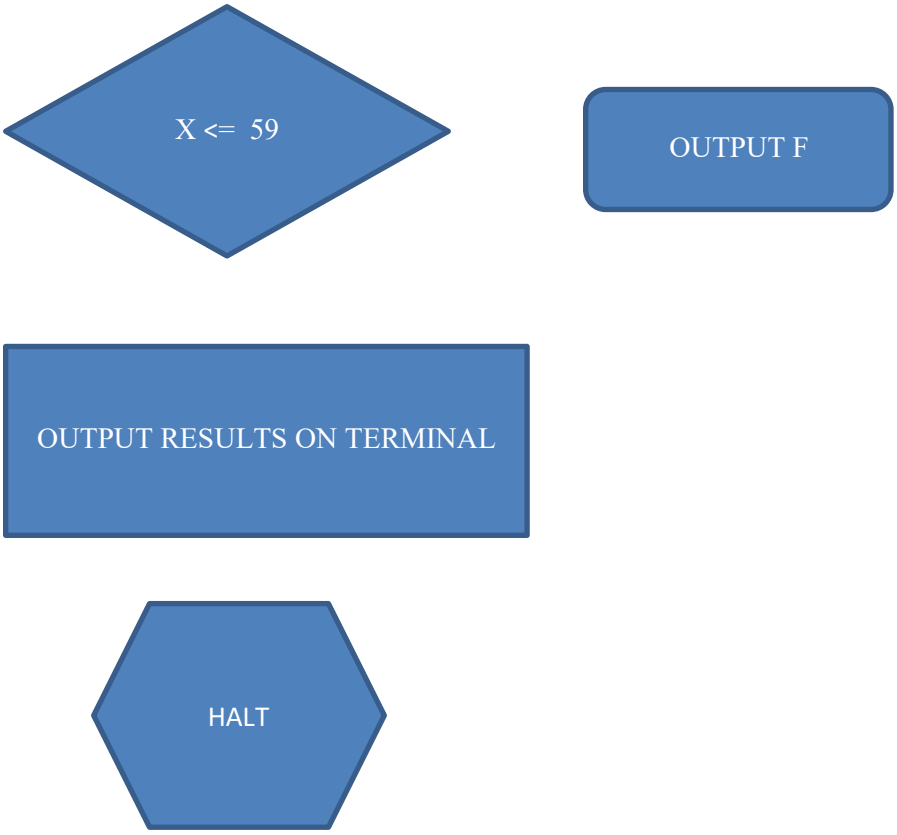
Supporting and background information may be appropriate to include as an appendix – things like results of user surveys, examples of problems to be solved by the applications, etc. Some of the supporting information may be graphical – remember all those charts you drew trying to explain your document to others?

Appendices can be used to address a specialized audience. For example, some information in the requirements document may be more important to the developers than to the users. Sometimes this information can be put into an appendix.

Flow chart or pseudo-code.

Include branching, iteration, subroutines/functions in flow chart or pseudocode.


START .ORIG x3000
User inputs scores
Valid Test score?
Calculate Minimum, Maximum, Average
90 <=X<=100
OUTPUT A
80<=X <= 89
OUTPUT B
70<= X <= 79
OUTPUT C
60<=X<= 69
OUTPUT D


X <= 59
OUTPUT F
OUTPUT RESULTS ON TERMINAL
HALT