CIS-11 Project Documentation Template

**Team Awesome**

**Luis Lopez, Jason Ash, Ana Moreno**

**Option B: Test Score Calculator**

**16 May 2025**

**Advisor: Kasey Nguyen, PhD**

# Part I – Application Overview

## Objectives

### The objective of this project is to design and implement an application for recording, processing, and analyzing student test scores. The system will:

### Automate the process of capturing student test scores.

### Calculate the average, maximum, and minimum scores for five test scores.

### Convert numerical scores to letter grades according to a defined standard.

### Improve accuracy, efficiency, and reporting compared to manual calculations

#### Why are we doing this?

### **Business Objective:** This application supports the institution’s goal of improving the reliability of grade reporting and reducing the manual effort required by faculty or teaching assistants.

### **Why now?** Current grade processing is time consuming, error prone, and lacks automation. With academic evaluation cycles always happening having a dependable grading tool is critical.

### **What if we delay?** Continuing with manual methods increases the risk of errors, late grade submissions, and dissatisfaction among students and faculty.

### **Who benefits?**

### Faculty – faster and more accurate grading.

### Students – more timely feedback and transparency in evaluation.

### Academic administrators – better reporting and auditability of grades.

## Business Process

### **Current Process (Without System)**

### Students submit assignments/exams.

### Instructors manually record grades in spreadsheets or paper logs.

### Averages and letter grades are calculated manually or using basic spreadsheet formulas.

### Grades are submitted to the institution’s learning management system or stored locally.

### **Proposed Process (With System)**

### Instructors enter raw scores into the grading application.

### The system:

### Converts scores to letter grades.

### Calculates statistical summaries (average, max, min).

### Results are displayed on-screen and optionally exported or printed.

### Grade reports are available for record-keeping and student feedback.

### **Key Benefits:**

### Reduces human error in grade conversion and calculations.

### Speeds up grading workflow.

### Standardizes how grades are determined and presented.

## User Roles and Responsibilities

### **Instructor**

### **Responsibilities:**

### Input student scores.

### Review calculated grades.

### Interpret statistical outputs (average, max, min).

### Report or export the final grades.

### **Tasks:**

### Enter up to 5 scores per session.

### Trigger calculation routines.

### Review automatic letter grade assignment.

### Use statistical output to analyze class performance.

## Production Rollout Considerations

### **Rollout Strategy:**

### Initial pilot with a small set of instructors for 1–2 courses.

### Collect feedback and iterate based on usage.

### Gradual expansion to other departments.

## Terminology

| **Term** | **Definition** |
| --- | --- |
| Score | Numerical value (0–100) entered for a student's performance. |
| Grade | Letter representation (A–F) derived from a score. |
| Average | Mean value of entered scores. |
| Max Grade | Highest score among all inputs. |
| Min Grade | Lowest score among all inputs. |
| PUTS / GETC | LC-3 system calls for output and input (in assembly context). |

# 

# Part II – Functional Requirements

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

## Statement of Functionality

### The grading application allows users to input exactly five numeric scores ranging from 0 to 100. Each score is entered via keyboard and is immediately validated to ensure it falls within the acceptable range. Invalid inputs are rejected, and the system prompts the user to re-enter a valid score. Once a valid score is accepted, it is converted from its ASCII representation into a decimal number and stored in memory.

### As each score is processed, the system assigns a corresponding letter grade based on a predefined grading scale. Scores between 90 and 100 receive an “A”, 80 to 89 receive a “B”, 70 to 79 receive a “C”, 60 to 69 receive a “D”, and scores below 60 are given an “”'. This grade is displayed on the screen immediately after the score is entered and processed.

### After all five scores have been entered and processed, the application automatically calculates the average score, identifies the highest (maximum) score, and determines the lowest (minimum) score. Each of these values is then converted into an ASCII representation and displayed to the user along with its corresponding letter grade. The application provides clear, spaced output lines for each result to ensure readability and clarity.

### Finally, the application halts its execution cleanly, signaling the end of the program. All processes from data entry to result output are conducted in a linear, user driven flow with immediate feedback after each step.

## Scope

### This application is designed to perform basic grading functions in a single use session. The initial version includes core features such as numeric score input, score validation, letter grade assignment, and calculation of average, maximum, and minimum scores. Each output is accompanied by a corresponding letter grade and displayed using a clear text format.

### 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.

## Performance

### The grading application is designed for fast and efficient processing. Each individual score is expected to be processed, converted to decimal, validated, graded, and displayed in under **0.5 seconds**. After the final score is entered, all calculations and their corresponding outputs are completed within **1 second**.

### Arithmetic operations are optimized for accuracy, ensuring correct addition, division, and comparisons within the valid score range of 0 to 100.

## Usability

### This application is designed to be simple and intuitive for instructors even with minimal technical knowledge. Input is handled directly through the keyboard, with each character copied to the screen to provide immediate visual feedback. This allows users to confirm that their input has been correctly received.

### The output is formatted for readability, with appropriate use of spacing and line breaks to separate different types of data. There are no complex commands, menus, or navigation layers as users follow a straightforward linear process from input to final results. The design ensures that the application can be comfortably used in classroom, lab, or home office environments.

### 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.

# 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** |
| 5/9/25 | Sort and display test grades in descending order to delineate which is the max and min. | Jason Ash | This might require a bubble sort and be beyond the scope of this project. | Low | Not yet started |
| 5/9/25 | Bubble short to display test grades in ascending order. | Jason Ash | This might beyond the scope of this project and is almost essentially doing project A as a sub-project for B. | Low | Not yet started |
| 5/9/25 | Allow for backspace while entering characters | Jason Ash | This would be nice from a user interface perspective, but will it take too much time to implement and is it expected? The LC-3 may not be capable of backspacing an entry since it only displays a box character in the place of a backspace instead of deleting the character to the left as expected by pressing the backspace key. | Low/ not possible | Not yet started |
| 5/9/25 | Invalid char function for both second and third digits without reentry of previous chars? | Jason Ash | This would be nice because if the user has a typo on the second or third digits, then the previous digit will not have to be reentered. This might not make since, though, because if an invalid character is entered for the second or third digit, it would be better for the user to just reenter that test score. | Low | Not yet started |
| 5/11/25 | Average more than five test scores | Jason Ash | This is probably beyond the scope of this project. | Low | Not yet started |
| 5/11/25 | Average other assignments in additions to tests. | Jason Ash | This is probably beyond the scope of this project. | Low | Not yet started |
| 5/11/25 | Averaging test scores for all of the students in a class instead of just for one student at a time. | Jason Ash | This is probably beyond the scope of this project. | Low | Not yet started |
| 5/11/25 | Combining the multiplication functions for \*10 and \*100 into one function (if possible) | Jason Ash | Since both functions do multiplication, is there a clever  way to just have one multiplication function? | Medium | Not yet started |
| 5/11/25 | Combining the hex negative 30 ASCII offset for the first second and third characters entered into one function. | Jason Ash | Since the hex ASCII offset to convert them to numeric digits is the same for all three characters, maybe one function can handle this for all of them. | Medium/Low | Not yet started. |
| 5/13/25 | Test averaging program correctly identify test scores over 100 instead of thinking the max is 199. | Jason Ash | Test scores should be from 0 to 100. Our program currently only displays an error if the test score is over 199. | High/critical | In progress |
| 5/13/25 | Test score program does not stop entry of test scores after five are entered. | Jason Ash | The test score program should only allow the user to enter five test scores and then do the processing and output according to its functional requirements. | High/critical | In progress |
| 5/13/25 | Fix multiplication subroutine(s) because if the test score is two or three digits, then the hex number placed on the stack is equivalent to a decimal number in the thousands. | Jason Ash | The test score program needs to put the correct hexadecimal equivalent of a test score entered on the stack for later processing. | High/critical | In progress |
| 5/13/25 | Figure out what happened in the most recent version and why it is only letting us enter one test score. | Ana Moreno | As stated above, the program needs to allow for entry of five test scores and progress from entering one tests score to the next in a way that is “natural” to the user. | High/ critical | In progress |

# Part III – Appendices

### The grading application will be developed to streamline the process of collecting and analyzing small batches of numeric test scores. The current system is particularly well suited for academic environments such as classrooms, tutoring centers, or training programs where quick feedback is essential.

## Flow chart or pseudo-code.

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