CIS-11 Project Documentation

**The Assemblers**

**Hailey Campbell, Muhammad Cheema, Christian Hong**

**Test Score Calculator**

**May 25, 2025**

**Advisor: Kasey Nguyen, PhD**

# Part I – Application Overview

## Objectives

### This project aims to provide teachers with an efficient, data-driven tool for analyzing student test scores. By automating score evaluation, the system will enable educators to gain deeper insights into class performance, reducing the time spent on manual calculations. Teachers will be able to enter student test scores, and the program will automatically compute the highest, lowest, and average scores, presenting the data in a clear and accessible format.

### Beyond saving time, the system will help educators identify trends in student performance, allowing them to pinpoint areas for improvement and adjust teaching strategies accordingly. Without this tool, teachers will continue relying on outdated methods, limiting their ability to make informed, data-backed decisions. Implementing this solution will not only streamline workflow for educators but also benefit students, as schools and administrators will be able to monitor academic trends and student success more effectively. The insights generated will contribute to refining educational approaches, ensuring students receive the necessary support to enhance their learning experience.

## Business Process

### Currently, teachers rely on manual methods such as spreadsheets, calculators, or paper-based systems to analyze student test scores. This process is time-consuming and prone to errors, making it difficult for educators to efficiently assess student performance and identify trends. Without automated tools, teachers must individually calculate averages, highs, and lows, which slows down decision-making and limits their ability to provide timely academic support. Additionally, administrators often receive incomplete or delayed reports, impacting their ability to track overall student success.

### With the introduction of this program, the process will become significantly more efficient. Teachers will be able to input student test scores into the program, which will instantly generate key performance insights, including highest, lowest, and average scores. The program will help educators identify trends in student achievement, helping them recognize areas where students may need additional support. By automating score analysis, the system reduces the time spent on manual calculations and minimizes errors, ensuring more accurate assessments. Furthermore, schools and administrators will gain access to comprehensive performance data, enabling them to refine educational strategies and improve student outcomes. This transition will not only streamline the workflow for teachers but also enhance the effectiveness of academic planning at all levels.

## User Roles and Responsibilities

### The system will be utilized by several users, each with distinct roles and responsibilities that contribute to the effective analysis of student test scores. Teachers are the primary users of the system, responsible for inputting student test scores and reviewing the program’s outputs to assess class trends. By analyzing the highest, lowest, and average scores, teachers can identify areas where students may need additional support and adjust their teaching strategies accordingly. Administrators will use the system periodically to monitor overall student success and evaluate educational effectiveness. They will rely on generated reports to track academic trends, assess teacher performance, and make informed decisions about curriculum development and resource allocation. While students are not direct users, they benefit from the insights provided by the system, as it enables more targeted instruction and improved learning opportunities. Additionally, school board members and policymakers can leverage aggregated data to shape education policies, allocate resources, and implement initiatives that enhance student achievement.

### The workflow begins with teachers entering student test scores, followed by system-generated analysis that informs decision-making at various levels. Administrators use these insights to refine educational strategies and improve institutional performance. By streamlining score analysis and ensuring data-driven approaches, the system enhances efficiency and accuracy in academic assessment across all roles.

## Production Rollout Considerations

### The rollout process will be structured in phases, beginning with a soft launch where a select group of teachers and administrators will test the system in a controlled environment. This phase will allow for early feedback, identification of potential issues, and necessary refinement. Once the system has been validated, a full rollout will be initiated across all intended users, ensuring that teachers, administrators, and school officials have access to its functionalities.

### To support the rollout, the system data must be properly populated. This includes importing historical test scores. The expected data and transaction volume must also be considered. The system will handle a significant amount of student test scores, requiring robust data storage and processing capabilities. Performance testing will be conducted to assess system responsiveness under peak usage conditions. Monitoring tools will be deployed to track system performance and address any technical issues promptly. This strategy will help ensure that teachers and administrators can seamlessly transition to the new system, maximizing its benefits for student performance analysis.

## Terminology

### **Test Scores** – Numeric values representing student performance on assessments.

### **Performance Metrics** – Statistical measures such as highest, lowest, and average scores used to evaluate student achievement.

### **Trend Analysis** – The process of identifying patterns in student test scores over time to assess progress and areas for improvement.

### **Data Input** – The action of entering student test scores into the system for automated analysis.

### **Automated Analysis** – The system’s capability to process test scores and generate insights without manual calculations.

### **Educational Insights** – Actionable information derived from test score analysis to support teaching strategies and student success.

### **Administrators** – School officials responsible for overseeing academic performance and decision-making based on system-generated reports.

### **Workflow** – The sequence of tasks performed by users to input, analyze, and interpret student test scores.

### **System Rollout** – The phased implementation of the program to ensure smooth adoption by teachers and administrators.

# Part II – Functional Requirements

## Statement of Functionality

### Prompt User for Input – The program will display instructions for the user to enter the test scores to be analyzed.

### Store Test Scores – The system will read and securely store the entered test scores in a stack for processing.

### Determine Highest and Lowest Scores – Using a linear search algorithm, the program will compare each test score against the current highest and lowest values. The first entered score will be initialized as both the highest and lowest, and subsequent scores will be evaluated. If a score is higher than the stored highest, it will replace the previous highest value. If a score is lower than the stored lowest, it will replace the previous lowest value.

### Calculate Average Score – The program will iterate through all stored test scores, summing them together and dividing by the total number of scores to compute the average.

### Generate Output – The system will display the highest, lowest, and average scores in a clear format. Additionally, it will assign a letter grade equivalence to each score using structured conditional logic (if-else statements) and will output it with the score

## Scope

### Phase 1: User Input

### Develop ability to prompt user for input.

### Allow the user to type 5 test scores to be analyzed.

### Read the data.

### Store the data in a stack.

### Phase 2: Determine Highest and Lowest Scores

### Set the first score as the highest and lowest.

### Loop through all the scores. If a score is higher than the stored highest, it will replace the previous highest value. If a score is lower than the stored lowest, it will replace the previous lowest value.

### Store the highest and lowest scores.

### Phase 3: Average

### Add all the test scores together.

### Divide by the number of test scores.

### Store the average score.

### Phase 4: Output

### Output the lowest score and use an if-else statement to find the letter grade equivalence and output it.

### Output the highest score and use an if-else statement to find the letter grade equivalence and output it.

### Output the average score and use an if-else statement to find the letter grade equivalence and output it.

### Phase 5: Testing and Debugging

### Test and debug the program making improvements as necessary.

## Performance

### Prompt User for Input – The prompt will appear on screen in less than 3 seconds of the program running.

### Store Test Scores – The test scores will be stored in the stack correctly.

### Determine Highest and Lowest Scores – The highest and lowest scores will be selected accurately.

### Calculate Average Score – The average score will be calculated accurately.

### Generate Output – The output will be shown within 5 seconds of the user entering the data.

## Usability

### The user will be able to enter 5 test scores to be analyzed.

### The lowest, highest, and average scores will be output in an easy to interpret format.

### New users should be able to complete basic tasks (inputting scores, viewing results) within 5 minutes of first use, with minimal instruction.

# 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** |
| 2025-05-22 | Integration with school student information  system | School  Administrators | Entering students' scores manually is still time consuming. Direct integration would automate data import, which saves teachers time. | High | V2.0 |
| 2025-05-22 | Goal setting and progress tracking | Students and teachers | It allows teachers to set learning goals for the students and helps show the data which tracks their progress towards those goals using the data. | Medium | v.2,1 |
| 2025-05-22 | Adding Customizable Reporting Dashboards | Administrators | It allows the users to customize their dashboard by giving selections of examples of visualizations they want to see on their dashboard. | Low | v2.2 |

# Part III – Appendices

## Pseudo-code

**Start of Program:**

Initialize stack pointer on top of stack

Initialize sum = 0

Initialize count = 0

**Main:**

JSR Read\_Scores

JSR Process\_Stack

JSR Calculate\_Average

JSR Display\_Results

Halt

**Subroutine Read \_Scores:**

Loop until count reaches 5

Prompt user for score as ASCII value

Convert ASCII to integer

Push score onto stack

Increment count

Return

**Subroutine Process\_Stack:**

Pop first score from stack

Set first score to min

Set first score to max

Set first score to sum

Count = 1

Loop until count reaches 5:

Pop score from stack

If score < min then update our min value

If score > max then update our max value

Add score to our sum

Increment count

` count – 5

If zero or positive branch to loop

Return

**Subroutine Calculate\_Average:**

Average = sum / 5

Store -5 in a register

Count = 0

Dloop

sum - 5

Increment count

` count – 5

If zero or positive branch to Dloop

return

**Subroutine Letter\_Grade:**

If score < 60 return ‘F’

(score – 60

If negative return ‘F’)

Else if score < 70 return ‘D’

(score – 70

If negative return ‘d’)

Else if score < 80 return ‘C’

(score – 80

If negative return ‘c’)

Else if score < 90 return ‘B’

(score – 90

If negative return ‘B’)

Else return ‘A’

**Subroutine Display\_Results:**

Output minimum score to user

Call Letter\_Grade and display the letter grade

Output maximum score to user

Call Letter\_Grade and display the letter grade

Output average score to user

Call Letter\_Grade and display the letter grade

Return