# GRADEBOOK
# 211 Project

Professor Jonathan Schrader

Nathaniel Brown

Rebecca Iselin

June 21, 2023

# Table of Contents

## Summary

The purpose of this project is to develop a tool that will allow a user to read a file as input and accurately calculate the student's current standing within their course. For the purposes of testing and relevance to this course, we decided to utilize the grading format and structure of CSC 212 according to the syllabus for the Summer of 2023. We wanted our program to be flexible and have the ability to take a variety of different input scenarios. Our program has the ability to read a 'completed semester' file in which there is a relevant grade for each of the categories within the course (i.e. assignments, labs, exams, projects). We also wanted to have our program adequately handle an incorrect number of data entries, i.e. if more than the expected number of grades are taken in, we will have measures to adjust that data. Lastly, we decided that a proper gradebook would allow for an incomplete data set to be read; and give the ability to update and fill the data set, while maintaining an ability to output properly.

Essentially the program will be able to intake a dataset containing a student's name and lines consisting of their grades in various categories within their course; this data can then be used to output a series of calculations specific to the user's requests. The types of calculations and outputs the program will undertake are determined by the user's selections within a basic menu user interface that our group constructed. An added feature of our program is the ability to edit data that was stored via our intake; and output accurately updated calculations upon request and output a rewritten file with the changes made to the data set inside our program's runtime.

## Planning

We started our project planning by meeting in person to discuss how we should go about the fundamentals of our project. We undertook the progression of our project development in a series of stages. After week 1 concluded, we had the barebones of a class structure set to produce an object in which a series of vectors represented members containing individual course category grades. We called this version of our program 1.0; as it was an extremely early version of production and was only effective via getters / setters to fill the data.

In version 2.0, we worked towards developing methods and helper functions as well as starting the process of creating our intake function to properly read the data. This was done by adding these functions into the members of our class that we had developed the week prior. Once we ensured that these methods were flexible and usable with any of the given vector members of our class type, we moved on to the final stages of development.

In version 3.0 we started to implement a basic user interface that allowed for the user to navigate through a series of menus and direct the output specific to the requirements of the project. These outputs include an individual return which outputs the student's name and grade in the course. A category menu allows for an output of every grade within a specific category (i.e. assignments) as well as an output of that specific average. Also, a 'course' menu which allows for an additional submenu output of essentially; a 'full report-card' for the student which outputs each category's grades as well as totals, strictly the category totals and course overall, and finally just outputting the course overall. These three menus were strictly chosen to display the way they are due to the requirements of the assignment description on github.

For our final version 4.0; we needed to implement the final objective of the 211 project: an ability to write output and write any changes back into the input file. For this we expanded our console to include a 'manual entry' sub-menu; which allows for the user to select a specific category and add any changes they would like to grades as well as include constraints to restrict the user to the expected number of grades per each category. After the successful implementation of this, we felt our project exceeded the needs of the project and could be considered finished.

# PseudoCode

## Version 1.0:

// – GRADEBOOK H/CPP –
// GOAL: Construct class - Class called 'gradebook' will be used to store vectors of doubles representing individual grades for assignments. The vector itself will represent individual categories within the gradebook object.
// Private Members should include: 4 vectors for each category; 4 integer variables representing the maxSize of each vector. The rationale for this being 'we're expecting 8 lab assignments, any more than that is invalid data'. (In a later version address validity on input intake.).
//Default constructor is the only needed constructor as in-take file will fill the empty vectors created upon construction.
// – MAIN CPP –
// Utilize CLA to store the string of txt fileat time of compile; for usage within intake function in a later version.
// Create empty obj of our class type.
// Manually fill vectors and ensure class is storing data as intended.
// In a later version; implement some form of a console for the user to navigate through the program. (Conditional based.)

## Version 2.0:

// Gradebook H/CPP file — (No changes in main at this time - aside from testing the methods / helpers)
// Develop two methods: 1 method will take any given vector and size; and calculate a specific average to that set of grades, the 2nd method will take the returns from the aforementioned method and output a current course grade.

// Specific Average Method - Passed three arguments, by reference we pass the vector containing grades for a specific category, we pass an integer representing the max number of grades that can be within the file, and a double representing the maximum possible score for this category.
// Create a sum and average integer
// Loop through each grade within specific vector
// add each grade to sum
// average set to (sum value / maximum grades) / maxScore.
// return the average

// Course Average Method - Passed 4 constant vectors; each of the specific category containers.
// set an int to our total points possible for the course (i.e. 1000)
// set doubles for points earned and our average
// series of conditionals following this format (for each of the vectors)

// if the vector is empty; remove the 'max points' for that section (i.e. assignments -200, exam -100), this will account  for no grades by deducting the appropriate amount of credit from the user.
// otherwise (else) create a for loop that will iterate through each grade in the current vector
// and add each grade to the pointsEarned total.
// After doing the above for each category vector; create a conditional if total points is greater than 0 (i.e. the student has submitted something) return the average set to points earned divided by total points, multiplied by 100.

## Version 3.0:

// Gradebook H/CPP
// Develop the barebones of a user interface to allow the user to navigate the program and direct output to the
// specific requirements of the project. I.e. allow the user to call methods on command and output appropriately.
// Using a conditional heavy system; of if / elif statements create a 'three tiered' structure.
// First create a variable for the outer user choice (i.e. menu directing to sub menus or termination of program)
// While user's input indicates they do not wish to terminate the program; select menu to open —>
// Before initiating any of the sub menus - create an 'inner user choice' that will reset upon returning to the top of this while loop (used for navigation within the sub menus - and flag for termination).
// For each of those menus; If the user does not wish to return to the main menu;  Select the task to undertake —>
// Output the task and redirect to inner menu for additional commands; until directed to return to main menu
// When given specific input - end outer while loop; allowing program to terminate.
///// Rough Outline /////
// Outer Choice = 0; // used to terminate outer while loop and direct conditionals
// While User Choice != 5…. Do —>
// Inner Choice = 0; // used to terminate inner while loops and direct conditionals
// If Outer Choice == 1… open submenu 1  (And so on through 4)
	// while Inner Choice != 5
// prompt user for submenu inner choice… Do —>
	// Related output for inner choice
// goes back to prompt until…
// Inner Choice is set to 5 (i.e. return to main menu
// initial while loop restarts; inner choice is set back to 0
// When user inputs outer choice == 5;
	// display termination message and exit Outer While loop
///// End Rough Outline ////

## Version 4.0:

// Modularize Code - reduced redundant repetitive calls and turned them into function calls
// General Revisions to output — doubles were printed with multiple digits; identified a 'tostring' output causing incorrect number of decimals wanted.
// Adjusted Calculations for averages that were not correct and fixed them to correctly calculate weighted averages.

//Manual entry
// checks for max number of grades — or partial entries
// cuts you off if too many; allows you to save regardless of how many parietal entries you enter
// rewrites the initial txt file
// input validation - checking to see if grade entered is a valid for respective category (i.e. grade size i.e not above 25 or whatever)

# Instructions for compiling

## Preliminary Compilation Instructions

The following commands must be run in the terminal in order to run the program:

- `g++ main.cpp Gradebook.cpp -o prog`
- `./prog test_full.txt`

The text file can be completely blank or already contain headers/data to intake.

```
Name Rebecca Iselin
Labs
Assignments
Projects
Exams 100
```

A file containing headers would look like this.

If the file to be read contains any data, grades must be delimited by one space.

Zero credit for assignments must be entered as zeroes to count towards a student's average.

## Running the program with blank input file

If the user is creating a brand new gradebook using a completely blank text file, the program will first prompt the user to input a student's name as shown below.

```
Please enter student name:
```

## Running the program with input file containing data

An input file containing data must follow the template shown above (see Preliminary Compilation Instructions).

Once the input file has been read, the user will be prompted to make a selection from the main menu:

```
1.) Manual Intake
2.) Individual: Report Student's Name and Overall Course Grade
3.) Category: Report Specific Category Grades
4.) Course: Full Course Report
5.) End Program
>>
```

## Manual Intake

The Manual Intake menu allows a user to manually input grades for a student. There are four subject categories: Labs(1), Assignments(2), Projects(3), and Exams(4). This menu allows the user to save(5) the entries made which are written back out the input file.

```
Manual Intake; Please input the number correlated with the category you would like to update:
1.) Labs
2.) Assignments
3.) Projects
4.) Exam
5.) Save Changes
6.) Return to Main Menu
>>
```

Once a category has been selected, grades can then be entered. After each grade, the user will be prompted to enter 'Y' if they have more grades to enter or 'N' if they have completed entries for that subject category.

```
>>1
Please enter Lab grade: 22
Enter Y if you have another grade to enter or N to return to menu
>>Y
Please enter Lab grade: 14
Enter Y if you have another grade to enter or N to return to menu
>>N
```

If a user attempts to enter a grade that is greater than the maximum allowed, an error message will display, and the user will be prompted to reenter a valid grade. If a user attempts to enter more grades than are allowed for the selected category, an error message will display and they will be redirected to the previous menu.

```
Please enter Lab grade: 26
Invalid grade entered. Max grade allowed is: 25
Please enter Lab grade:
```

```
Max number of lab grades have been entered.
```

Once the maximum number of grades have been entered, a message will display notifying the user and the Manual Intake menu will redisplay allowing the user to continue adding grades for the remaining subjects, save their changes, or return to the main menu.

To save changes to the input file, enter 5 and a message will display that the changes were successfully made.

```
------------------------------
Data saved to file
------------------------------
```

Enter 6 to return back to the main menu.

### Project Grades

When a user elects to enter project grades, they must specify which project grade to enter from the project grade entry menu. The user's selection will determine the validity checks performed on their input.

```
Which Project Grade Would You Like To Add?
NOTE: you must ensure you are modifying the correct Project values listed below to maintain accuracy of report.
Failure to do so is your own doing.
Only update 1 Project1 grade and 1 Project2 grade
1.) Project 1: (Maximum Points 150)
2.) Project 2: (Maximum Points 350)
3.) Return to Previous Menu
>>
```

Entering 1 will allow the user to enter a grade for project 1. Entering 2 will allow the user to enter a grade for project 2. Entering 3 will return the user to the previous menu where they may continue entering grades for other categories.

```
1.) Project 1: (Maximum Points 150)
2.) Project 2: (Maximum Points 350)
3.) Return to Previous Menu
>>2
You have selected Project 2. Enter a grade between 0 and 350:
371
This is not a valid input... input a grade between 0 and 350:
342
```

## Individual: Report Student's Name and Overall Course Grade

Option 2 from the main menu will display the student's name and overall course grade at that point in time. The course average will be reflective of the grades entered and missing grades will not lower the student's score.

```
Individual Output:

Student's Name: Rebecca Iselin
Student's Course Grade: 72%
```

## Category: Report Specific Category Grades

Option 3 from the main menu will display a submenu allowing the user to generate reports based on a selected subject category.

```
Select Specific Category; Please look at the menu of choices and input the number correl
ated to your choice:

1.) Report all Assignment Grades and Total
2.) Report all Lab Grades and Total
3.) Report all Exam Grades and Total
4.) Report all Project Grades and Total
5.) Return to Main Menu
```

Each report will display individual grades and the average of those grades. Averages are calculated based on the grades entered. Only grades entered as zeroes will count against average.

```
Lab Grade List: 25 24 22 20 19
Lab Average: 88%
```

## Course: Full Course Report

Option 4 from the Main Menu allows a user to generate a full course report for a student. This will display a submenu with three report options: full report containing all grades and averages, a report card containing only averages, or course average only.

```
Course; Please look at the menu and input the number for which full report you would like.

1.) Full Report Card and Course Grade (Outputs Every Grade)
2.) Report Card: Averages of All Categories and Course Grade
3.) Course Average Only
4.) Return to the Main Menu
```

The full report card and course grade will generate a report containing all of the student's individual grades, category averages and overall course average. While category averages are reflective of grades entered at that point in time, the course average is reflective of all current averages.

```
Full Course Report Card: (Every Grade, Category Averages, and Course Average)

------------------------------
Lab Grade List: 25 24 22 20 19
Lab Average: 88%

------------------------------
Assignment Grade List: 48 42 50
Assignment Average: 93%

------------------------------
Project Grade List: 148
Project Average: 99%

------------------------------
Exam Grade List:
Exam Average: 0%

------------------------------
Class Average: 72%
------------------------------
```

Option 2 will display a report card listing current averages for each category and overall course average.

```
Report on Student: Averages for Each Category and Overall:

Lab Average: 88%
Assignment Average: 93%
Project Average: 99%
Exam Average: 0%
------------------------------
------------------------------
Class Average: 72%
------------------------------
```

Option 3 will display a report containing only the course average.

```
Class Average Only

Class Average: 72%
------------------------------
```

# Member Contributions

## Version 1.0

| Version 1.0 | Tasks: | Notes on Task: | Start Date: | Initiated By: | End Date: | Finished By: | Notes: |
|---|---|---|---|---|---|---|---|
| | Meeting 1: Planning | Meeting to map out the overall flow of the project; and brainstorm various ideas and paths we could take to complete the projects objectives. | 5/30 | All Members | 5/30 | All Members | Upon conclusion of the meeting; the group unanimously decided to make the barebones of a class; that would best represent their idea on how to tackle the task. Going Forward. Next meeting will determine what tasks need to be created and undertaken to progress. |
| | Meeting 2: Class Construction and Discussoin on Usage of Members / Methods | Compare Barebones Class h/cpp and discussion on usage of methods and members in relation to barebones creation. | 5/31 | All Members | 5/31 | All Members | Decided on which file to use for version 1.0 and established tasks to be undertaken throughout the week. |
| | Intake Function: | Create a function to read the given txt file and sort the data into respective category grade vectors of oru created object. | 5/31 | Nathaniel Brown | 6/2 | Nathaniel Brown / Rebecca Iselin | Initial Function developed with barebones; class code. Debugged and improved by Rebecca. |
| | Main File Creation: | Create instance of our object of type gradebook and store variables for CLA (i.e. txt file name) | 5/31 | Both - In Meeting | 5/31 | Both - In Meeting | Simple main file to work with in the future; containing a CLA variable for our txt input at argv[1]. Created by both members. |

## Version 2.0

| Version 2.0 | Tasks: | Notes on Task: | Start Date: | Initiated By: | End Date: | Finished By: | Notes: |
|---|---|---|---|---|---|---|---|
| | Meeting 3: Priority Methods | Meeting to discuss the main methods used within our program; i.e. calculate averages for the categories and the course grade. | 6/1 | All Members | 6/1 | All Members | Upon conclusion we established tasks to undertake and test throughout the week; and set up next meeting date for the following day. |
| | Meeting 4: Work | Compare Barebones Class h/cpp and discussion on usage of methods and members in relation to barebones creation. | 6/1 | All Members | 6/1 | All Members | Majority of work started on two methods for project - to be completed throughout week. Additional goal of starting template for next week's version. |
| | Calculate Category Average: | Create a method to take a filled vector and return the average of that specific category. | 6/2 | Rebecca Iselin | 6/2 | Rebecca Iselin | Finished method to calculate category averages and return specific average |
| | Calculate Course Average: | Create a method to take the returned averages from Category average. Use these to calculate the course average for function. | 6/2 | Nathaniel Brown | 6/2 | Nathaniel Brown | Finished method to calculate course average using above values and return course average |
| | Template Menu Code: | Create generic template for conditional based menu; for a future implementation. | 6/3 | Nathaniel Brown | 6/4 | Nathaniel Brown | Generic Template for future use complete; menu of sub menu's containing more submenus... Temporary output. |

## Version 3.0

| Version 3.0 | Tasks: | Notes on Task: | Start Date: | Initiated By: | End Date: | Finished By: | Notes: |
|---|---|---|---|---|---|---|---|
| | Meeting 4: Menu Implementation and Future Task Planning | Meeting to discuss implementing template menu and expanding upon what we have. I.e. Generate more tasks to keep the project moving. | 6/6 | All Members | 6/6 | All Members | Started to implement Menu; set goals for a 'Manual Intake' section of the code to adequately satisfiy criteria of assignment. Finalize variations of TXT files for flexibility testing. |
| | Meeting 5: Work | Meeting to finalize TXT file contents and test cases. Initiate work on taks for the week. | 6/7 | All Members | 6/7 | All Members | Decided on TXT File cases for: Full dataset, Partial Dataset (Display CURRENT averages; not as if they have 0s for the remaining grades), and an empty dataset (Future; manually fill in the grades and write to a fill). |
| | Manual Intake: | Create a function to later be implemented into menu; allows for the user to manually update and fill vectors if the txt file is not a complete gradebook for the student. | 6/7 | Rebecca Iselin | 6/9 | Rebecca Iselin | Finished ability to manually intake; ability for validation checks and writing files from empty -> full will be done next week. |
| | Implement Menu: | Create a method to take the returned averages from Category average. Use these to calculate the course average for function. | 6/7 | Nathaniel Brown | 6/9 | Nathaniel Brown | Finished general implementation of main menu, which can call various methods and output appropriately depending upon user request. Has basic validation setup for if users input is out of range of the expected parameters (with related error message). Needs to have Manual Intake added. |
| | TXT Finalization: | Finalize and upload TXT files into github; so we are working with the same test cases. Both create two. | 6/7 | All Members | 6/7 | All Members | Generic Template for future use complete; menu of sub menu's containing more submenus... Temporary output. |

## Version 4.0

| Version 4.0 | Tasks: | Notes on Task: | Start Date: | Initiated By: | End Date: | Finished By: | Notes: |
|---|---|---|---|---|---|---|---|
| | Meeting 6: Start Report Work - Recap Work Done Over The Weekend | Discuss writing of the report and its requirements; divide into equal portions. Finish implementing changes and additions made over the weekend. | 6/12 | All Members | 6/6 | All Members | Started tasks for finishing the report. |
| | Meeting 7: General Testing and Hotfixes | After confirmation from Professor; start recording process for submission. | 6/14 | All Members | 6/14 | All Members | Worked on troubleshooting manual intake conditionals for specifically the project grades as there was no way to determine between 150/350 point limits. Uploading this weeks changes as version 5.0. |
| | Weekend Additions: | Additions to manual intake. | 6/10 | Rebecca Iselin | 6/10 | Rebecca Iselin | Added validation to manual intake; to ensure that vector size does not exceed the limit set by the course. As well as rewriting functionality when the user opts to save their changes. Overwrites the TXT file used during compile. |
| | Report - Summary: | Summarize the overview of the project; break down mini summaries of each version of production. | 6/12 | Nathaniel Brown | 6/12 | Nathaniel Brown | Finished the Summary of various versions of the project. |
| | Report - Instructions / Compile | Detailed Instructions and guide for compiling, work through individual test cases and appropriate error messages / what the program does in those cases. | 6/12 | Rebecca Iselin | Ongoing - as code continues to develop. | Rebecca Iselin | See initial note. |
| | Report - Spreadsheet | Spreadsheet has been slowly being filled and reogranized since initial meeting on paper; migrated to google sheet for submission. | 6/12 | Nathaniel Brown | Upon Submission - WIP | Nathaniel Brown | See initial note. |
| | Report - Pseudo Code | Create summary and individual breakdown of pseudo code for various stages of our projects progressoin. | 6/12 | All Members | 6/12 | All Members | Finished in Meeting 6: As both of us had our own individual pseudocode, we melded them together for the report. |

## Version 5.0

| Version 5.0 | Tasks: | Notes on Task: | Start Date: | Initiated By: | End Date: | Finished By: | Notes: |
|---|---|---|---|---|---|---|---|
| | Meeting 8: | Meeting to troubleshoot minor calculation bugs; as well as set up our recording process. | 6/20 | All Members | 6/20 | All Members | By the end of this meeting we established the remaining tasks to be completed before our recording and submission |
| | Meeting 9: | Meeting to peer review report and instructions and record the final project video portion of the assignment. | 6/21 | All Members | 6/21 | All Members | Finished! |
| | Testing Menus: 1-3 | Ensure menu's work for all possible data input scenarios. Range, Data Type, Varying Scenarios, Ect. | 6/20 | Rebecca Iselin | 6/20 | Rebecca Iselin | Done |
| | Testing Menus: 4 | Ensure menu's work for all possible data input scenarios. Range, Data Type, Varying Scenarios, Ect. | 6/20 | Nathaniel Brown | 6/20 | Nathaniel Brown | Done |
| | De - Modularizing Menus: | Initially modularized menu's into seperate function calls - redundant and edited back to its inital state. See end notes. | 6/20 | Rebecca Iselin | 6/20 | Rebecca Iselin | Due to the fact that our menus function within while loops; the code is not being rewritten multiple times even though the menus display multiple times. Unanimously we decided to revert to a previous iteration of our menu's to maximize readability of our code. Minor modularization of inner menu's that were repeated. |
| | Refine Project Calculations | Specific issue regarding calculation of project grade average when only 1 project grade has been read so far. | 6/20 | All Members | 6/20 | All Members | Essentially added member variables to store the specific grades to project 1 and |
| | Finalize Instructions: | Need to finalize instruction menu after the completeion of calculations. | 6/21 | Rebecca Iselin | 6/21 | Rebecca Iselin | Word Document created for Instructions; transferred old google docs info into the word file for submission. |
| | Update Pseudo Code | Format and adjust psuedocode for the entire document for readability | 6/20 | All Members | 6/20 | All Members | We each ran through seperate sections of the project and filled in appropriate pseudo code / reorganized the flow of helpers, methods, getters, setters, ect to make the program more readable for the user. |
| | Finalize Report: | Once Instructions are completed - format document for peer review and submission. | 6/22 | Nathaniel Brown | 6/22 | Nathaniel Brown | Finished during meeting; will need the last image of our contribution table immediately after recording finishes. |
| | Recording: | Once Code and Pseudo are implemented fully; begin recording for submission. | 6/22 | All Members | 6/22 | All Members | Recording finished; ready for submission after we review the file. Now to start Project 2... |