# CS210 8-1 Journal: Portfolio Submission.

In this project, the objective was to collaborate with Airgead Banking working with a local high school to develop a program that will teach students the concepts of fiscal responsibility (such as living within their means and spending less than they make) via an interactive system. The initial focus for this project will be on investing and the power of compound interest.

My task: Develop an application that allows users to see how their investments will grow over time. Airgead Banking has provided a list of functional requirements that describe what they need their application to do.

**Directions were as follows:**

1. Create pseudocode or a flowchart to plan my coding project. Outline my code step-by-step in order to use it as a guide when coding.
2. Write my thoughts in English of what the program should do.
3. List statements, each describing a single action (without use of syntax).

List all steps.

1. Using proper naming conventions.

Only one statement per line.

1. Developed an object-oriented programming (OOP) application using secure and efficient C++ code.

**The things I did well in this project were as follows:**

* I paid special attention to ensuring the application:
* Met all specifications listed in the Airgead Banking App Functional Requirements
* Followed best practices described in the Airgead Banking Standards document
* Included in-line comments

I think I found it a bit challenging to compartmentalize my code. In other words dividing my code into libraries even though the functionality was common, was challenging at first but I very quickly figured it out and eventually realized how it helped in making the process of programming more manageable and potentially reusable for me.

I did not find any piece of code to be particularly challenging more than I did syntactical errors. In Python, for example, I sometimes missed return statements hence not ending the execution of certain functions or in certain cases, default values were implicitly returned. I did not have that issue with the Airgead Banking App.

I think being able to write pseudocode and/or creating (process) flowcharts and then turning that into workable code or program was the biggest skill I learned from this project. I think being able to have visual representation or a systematic understanding of what it is the end user is trying to achieve, and being able to represent the users’ idea(s) in a diagram is just as important as being able to write the right code. Following standards was also one key lesson I learned in this project.

I think compartmentalizing code is key to making it maintainable as well as readable. Also using System (“CLS”) when needed helps tidy things up a bit. Separating certain components (CPP, H, and any other files used) in this project made a lot of difference in my opinion.