# **CS 499 Module Three Journal:**

**Marketing With ePortfolios and Artifact Update**

Andrew Emilio DiStefano

SNHU

Professor Neil Kalinowski

May 22, 2025

**Part One:**

1. **How might you use an ePortfolio for the benefit of self-promotion?**

Based on my experience placing professionals as a robotics software engineer recruiter, showing potential employers an ePortfolio of relevant projects is the single most powerful tool for securing a job as an out-of-college professional. I have personally seen candidates with only academic and internship software engineering experience secure positions with companies looking for someone with three to five years in the industry. Showing relevant projects is important here. For instance, having a collection of impressive full-stack web development projects will not be of much use for a robotics position. That being said, as long as the desired position is relevant to the projects in an ePortfolio, such a portfolio is an absolute game changer when applying for software engineering jobs.

1. **Describe possible downsides or risks—for instance, the risks of posting intellectual property online for public consumption.**

There are several risks associated with making an ePortfolio publicly available online. The first risk which comes to mind is of course the potential for one’s work to be plagiarized by dishonest engineers and developers in order to use in their own projects without giving one credit for one’s work. Another considerable risk in making software public on a platform like GitHub is the potential for malicious actors to perform penetration tests on software products downloaded to their systems. Leaving one’s code for any and all to see allows these malicious individuals to study one’s code and search for vulnerabilities. Dedicated cybercriminals could even develop an understanding for one’s coding style and common coding errors in order to penetrate other systems designed and built by the targeted developer or engineer, which is an alarming notion to say the least. Unless this is an active open-source project with developers and engineers around the world constantly improving system security, this could certainly lead to the compromise of the systems being used in one’s ePortfolio.

1. **How might you mitigate risks while maximizing the marketing potential of the ePortfolio?**

One way to mitigate the aforementioned risks associated with the public display of one’s ePortfolio is the restriction of the *Download Zip* option on GitHub. This can be accomplished from the developer settings, where developers have the option of disabling this feature. Another effective option would be to set the repository as private. This would make it difficult for potential employers to find one’s work online, unless of course one were to give an access key to potential employers to be reset at a later date. This way a select few people and organizations are able to view the ePortfolio without it being open to the entire internet. Changing the access key periodically mitigates against the risk of potential employers, or of malicious actors successfully penetrating the systems of potential employers, from using one’s projects maliciously.

1. **Which course outcomes have you achieved so far, and which ones remain?**

The work that I have done so far on the stated objectives has been consistently documented and commented. I have kept a comprehensive list of the libraries and dependencies which would be required for the PlantSitter system to work on another Raspberry Pi circuit, and I have compartmentalized my MemoryBot code into a series of functions which could easily be reused in other projects. I have been using the same names for my functions as those of similar functions built into the Python language. Since I am in the process of porting MemoryBot to C++, it helps to be able to call upon similar functions as those in the Python version while developing the C++ version of the system. I was even thinking of duplicating more of the functions built into Python in the C++ language and perhaps releasing the library to help Python developers overcome the learning curve associated with learning C++ as someone who has more experience with Python. The enhancements which I have made so far closely align with the course outcome of designing, developing, and delivering professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts. The way that these enhancements are being implemented makes it easier for a wider range of professionals to elaborate on the PlantSitter system, which of course aligns with the course outcome of employing strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.

In order to enhance system efficiency and to limit the complexity of the state machine logic used in the PlantSitter system, I have decided to add multithreading to this project. I will have one state machine responsible for temperature control running concurrently alongside another state machine to control soil humidity. This decision was made after carefully considering different system design options in order to meet the requirements in alignment with the course outcome of designing and evaluating computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices. The implementation of Python’s *threading* library also aligns well with the course outcome of demonstrating an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.

I have already started writing the security functions for the new and improved C++ version of the MemoryBot program. The functions which I have so far validate user input by checking for unreasonably long input strings, but I will also be implementing other standard C++ security measures as well. These security functions align with the course outcome of developing a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.

The work that I have done so far in completing the enhancements stated in the last module meets all of the course outcomes, but I certainly have more work to do in my completion of these enhancements to more fully achieve each of the course outcomes.

**Part Two:**

### **Status Checkpoints for All Categories**

| **Checkpoint** | **Software Design and Engineering** | **Algorithms and Data Structures** | **Databases** |
| --- | --- | --- | --- |
| **Name of Artifact Used** | PlantSitter | MemoryBot | PlantSitter |
| **Status of Initial Enhancement** | I have completed about half of the physical circuitry for this project, I added a breadboard to space everything out including the new buttons and the new LCD screen. I have not put on the LCD screen yet since I am still looking into how to best use the I2C protocol to connect the second screen to the same GPIO pins that are connected to the first screen.  I have imported *Thread* from Python’s *threading* library. I am currently in the process of setting up the functions which implement my two separate state machines in order for them to run concurrently. | I am proud to say that I have almost finished porting MemoryBot to C++! I still have a lot of work to do in order to successfully implement the Binary Search Tree with the inOrder search function rather than the iterative vector which is currently being used. I decided to include the vector data type just to get the system up and running, but I will be focusing on creating the BinarySearchTree as my next development with this artifact as soon as I am able to completely replicate the behavior of the original (still some logical bugs to iron out). I am really excited about this one! | I have installed the ReductStore dependency to enable database functionality for this project. I have been experimenting with this powerful database software and implementing simple programs to gain a firm understanding of how the database (which is meant to run in a docker container) is used in Python. I feel confident that over the next few days I can develop a function which will take values passed in by the temperature and soil moisture sensors. The function will arrange data into the ReductStore database so that it can be processed effectively. |
| **Submission Status** | Pending | Pending | Pending |
| **Status of Final Enhancement** | Improvements being made | Improvements being made | Improvements being made |
| **Uploaded to ePortfolio** | Yes | Yes | Yes |
| **Status of Finalized ePortfolio** | 70% Finished | 70% Finished | 20% Finished |