**James Galbreath**

**CS-499**

**7/20/25**

**1. Briefly describe the artifact. What is it? When was it created?**  
The artifact is a C++ command-line course planner that allows users to load, view, and search academic course data, including course numbers, titles, and prerequisites. The program reads data from a text file and organizes it using a Course class and a vector data structure. It was originally developed in an earlier SNHU course as a final project focused on data structures and algorithms in September of last year.

**2. Justify the inclusion of the artifact in your ePortfolio.**  
I selected this artifact for my ePortfolio because it demonstrates my ability to work with structured data, apply object-oriented design, and implement core algorithms in C++. It shows my understanding of vectors, file input/output, and searching techniques. The enhancements made for CS-499 further improved the artifact by adding sorting functionality based on user input, increasing interactivity, and making the code more modular and well-commented. These improvements showcase my growth in clean code design, user-centered functionality, and problem-solving.

**3. Did you meet the course outcomes you planned to meet with this enhancement in Module One?**  
Yes, this enhancement meets the planned course outcomes related to applying data structures and algorithms, analyzing problems and implementing solutions, and demonstrating software engineering principles. The added sorting functionality and enhanced user interface reflect deeper understanding and practical application. At this time, I have no changes to my outcome-coverage plan.

**4. Reflect on the process of enhancing and modifying the artifact.**  
Enhancing this artifact taught me how small UI changes—like adding a menu for sorting—can significantly improve usability and user control. I also gained experience with modularizing logic (by separating sorting functions), using lambda expressions for custom sorting, and strengthening code readability through comments. A key challenge I faced was designing a clean, intuitive interface without overcomplicating the program logic. Balancing simplicity with flexibility pushed me to think like both a developer and a user, which improved the final result.