Briefly describe the artifact. What is it? When was it created?

The artifact I selected is a course catalog application originally developed in C++ for CS300: Data Structures and Algorithms in April 2024. It reads a course list from a CSV file and allows users to display all courses and search for specific courses. The program uses a binary search tree to manage course data efficiently, supporting quick searches and maintaining ordered traversals.

Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?

I chose this artifact because it clearly reflects my abilities in software design and engineering, and it's a project where I could demonstrate meaningful enhancements. Originally, it was a C++ program that managed course information using a binary search tree. While functional, it had limitations in readability, modularity, and cross-language accessibility. Porting it to Python allowed me to not only translate the logic but also improve the structure and maintainability of the program.

Through this enhancement, I focused on several core aspects. First, I ensured the code was modular and organized, with clear classes and methods that separate concerns logically. The BST implementation was restructured to make traversals, insertions, deletions, and searches more intuitive, while also integrating validation for prerequisites. Input and output handling was refined to improve user experience, and I incorporated colored outputs and formatted tables to make the console interface cleaner and more professional.

I also paid close attention to error handling and edge cases, such as duplicate course IDs or missing prerequisites, which weren't addressed in the original C++ version. By enhancing these areas, I demonstrated not only technical proficiency in Python but also thoughtful design practices and attention to detail. This artifact now showcases my ability to analyze existing software, identify areas for improvement, and implement robust, maintainable solutions. It reflects both my technical skills and my capacity to think critically about program structure and usability.

Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

This enhancement met the course outcomes I had planned in Module One.

Specifically:

- Outcome 1: By structuring the program with clear classes and methods, I created a modular codebase that could support collaborative development.
- Outcome 2: Adding comments, structured functions, and clear console output demonstrates professional-quality communication of technical ideas.
- Outcome 3: Re-implementing the BST in Python allowed me to design and evaluate computing solutions using algorithmic principles.
- Outcome 4: Leveraging Python's built-in data structures and modules highlighted my ability to select effective tools for software solutions.

Overall, the enhancements reinforced my ability to translate design choices into functional, maintainable code, which aligns with the course outcomes.

Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

Working on this enhancement mostly reinforced skills I've been developing throughout my coursework, rather than introducing entirely new concepts. Porting the program from C++ to Python required careful thought about design decisions I might usually take for granted, like structuring the BST, handling file input/output, and validating prerequisites. Some parts translated easily, while others needed alternative approaches to fit Python's style and maintain correctness.

My main challenge was finding the right solution for parts that didn't have a direct Python equivalent. Experimenting with different methods and debugging them until they worked as intended really strengthened my problem-solving skills. It also highlighted the importance of testing and verifying functionality at each step.

Overall, this process reinforced my understanding of object-oriented programming, data structures, and program design. It wasn't just about moving code from one language to another, it was about thinking critically, applying best practices, and ensuring the program remained efficient and reliable.