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 algorithmic design, data structure optimization, and C++ programming, and it offers an opportunity to demonstrate meaningful enhancements. The conversion of the BST into an AVL tree significantly improves the performance of the application by ensuring O(log n) time complexity for insertions, deletions, and searches.

Through this enhancement, I focused on several key areas. First, I implemented AVL-specific balancing mechanisms, including left and right rotations, double rotations, height tracking, and balance factor calculations. These additions ensure that the tree remains balanced after all operations, which greatly improves efficiency and reliability.

I also improved code organization by introducing modular helper functions such as rotateLeft(), rotateRight(), and updateHeight(). These changes make the code easier to read, maintain, and extend. Additionally, I implemented a debug mode to display AVL balancing steps, aiding transparency and understanding of tree operations.

These improvements showcase my ability to apply advanced algorithmic concepts, structure code for maintainability, manage memory safely, and document logic clearly. This artifact now better demonstrates both technical skill and thoughtful design, making it a strong addition to my ePortfolio.

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 planned in Module One. Specifically:

- Outcome 1: By structuring the AVL implementation with modular functions and clear logic, the code supports collaboration and future updates.
- Outcome 2: Adding inline comments and debug output communicates the technical logic and AVL balancing process effectively.
- Outcome 3: Refactoring the BST into an AVL tree demonstrates the ability to design and evaluate algorithmic solutions for efficiency and correctness.
- Outcome 4: Implementing rotations, height tracking, and rebalancing demonstrates mastery of advanced algorithmic techniques.
- Outcome 5: Applying careful memory management and structured exception handling improved program reliability and security.

There are no major updates to my outcome-coverage plans, but this enhancement strengthened my emphasis on efficiency, maintainability, and reliability as central themes in my ePortfolio.

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 reinforced my understanding of balancing theoretical knowledge with practical implementation. Converting the BST to an AVL tree required revisiting concepts such as balance factors, rotations, and height calculations. Implementing these features tested my ability to translate algorithmic concepts into working code while maintaining the integrity of the original program.

A major challenge was ensuring that balancing logic correctly handled all cases, particularly during deletion operations where rebalancing can be complex. Debug mode proved invaluable for tracking operations and verifying accuracy. Another challenge was maintaining clarity and maintainability while introducing these algorithmic changes. This process strengthened my skills in algorithm design, debugging, code organization, and documentation. It reinforced the importance of balancing performance improvements with clean, maintainable code, a principle I will continue applying in my ePortfolio and future projects