**Artifact Description**

The artifact in question is a Binary Search Tree (BST) implemented in C++, designed to manage and organize bid data efficiently. Initially, this BST provided basic functionality for adding, searching, and removing nodes, each containing bid information such as bid ID, title, fund, and amount. The original version of the artifact was created as part of a SNHU CS300 course project, focusing on data structures and their applications in real-world scenarios.

**Justification for Inclusion**

This artifact was selected for inclusion in my ePortfolio due to its demonstration of fundamental software engineering principles, particularly in the area of data structures. The BST exemplifies efficient data management and manipulation, which are critical skills in software development. Specific components of the artifact that showcase my abilities include:

Implementation of BST operations: Adding, searching, and removing nodes effectively.

Enhancement with AVL Tree features: Improving the balance of the tree to optimize search times and maintain efficiency.

The enhancement involved transforming the basic BST into a self-balancing AVL Tree, which ensures that the tree remains balanced with operations staying efficient, even as data scales.

**Skills Demonstrated Through the Enhancement**

The primary skills demonstrated through this enhancement include:

Algorithm Optimization: By integrating AVL balancing, I showcased my ability to optimize algorithms for better performance.

Problem Solving: The process required thoughtful consideration of how unbalanced trees affect performance and devising a strategy to mitigate this through tree rotations.

Programming Proficiency: Effective use of C++ to implement complex data structures and algorithms.

**Reflection on the Enhancement Process**

Learning Outcomes: During the enhancement, I deepened my understanding of AVL trees, including how rotations help maintain tree balance and how this affects overall algorithmic efficiency. I also refined my C++ programming skills, especially in managing dynamic memory and pointer operations.

Challenges Faced: One of the main challenges was ensuring that the tree remained balanced after every insertion and deletion without significantly impacting performance. Implementing the rotation logic correctly required careful attention to detail and debugging.

Incorporating Feedback: Feedback from testing and peer reviews highlighted areas where the search function could be optimized, and how the removal process could be made more efficient. This feedback was crucial in refining the implementation to ensure the tree's balance and performance were maintained.

Improvements Made: The artifact was significantly improved by ensuring operational times for insertion, deletion, and search were kept to a minimum, regardless of the tree's size. This made the tree more suitable for applications requiring high efficiency in data management.

**Course Outcomes Met**

The following course outcomes were either partially or fully met through this enhancement:

Building collaborative environments: While the project was individually based, the enhancement required reviewing existing literature and community solutions, engaging with broader knowledge on AVL trees.

Delivering professional-quality communications: The process improved my ability to document and explain complex technical implementations clearly and concisely.

Designing and evaluating computing solutions: The enhancement directly involved evaluating the existing BST solution and designing a more efficient AVL Tree-based approach.

Using innovative techniques and tools: Implementing the AVL rotations and balancing techniques demonstrated innovative problem-solving in a common data structure application.

Developing a security mindset: Although not directly applicable to security, the meticulous attention to detail and consideration of all aspects of the tree's operations align with the thoroughness required in security-minded practices.

**Outcomes Not Fully Met**

Developing a security mindset: The project did not explicitly involve security aspects, such as data protection or secure coding practices, which could be areas for future exploration.

This reflection not only showcases the technical skills applied but also highlights the continuous learning process inherent in software development. The project was an invaluable exercise in applying theoretical knowledge to practical, real-world problems.