Artifact Description:

The artifact that has been selected for this milestone is a C++ advising assistance program, also notably known as, Code Two. This program enables users to load course data from a file, sort courses alphanumerically by course number, and display course details and prerequisites through a simple menu-driven interface. The program was initially created during [insert course name or project] as a demonstration of applying software engineering principles to a practical task.

My Justification:

The artifcat was selected for my ePortfolio because it is able to effectively demonstrate my abilities in software design, engineering principles, and the ability to deliver a functional, user-focused application. These are the specific components that could assist me in doing so:

- File Parsing & Data Processing: the program reads and parses data files into structured objects, demonstrating my ability to handle real-world data processing tasks.
- Sorting & Display Functionality: the implementation of std:: sort and a menu-focused interface highlights my understanding of algorithm selection and user experience design.
- Error Handling & Modularity: the enhanced code reflects my ability to identify and resolve issues, improve maintainability, and ensure robust functionality.

These aspects of the artifact align with the goals of showcasing proficiency in software engineering and design while adhering to coding standards and best practices.

Improvements Made:

Enhancements to this artifact focused on addressing weaknesses identified during the code review.

The improvements include:

- Strengthened Error Handling: Added checks for invalid file paths and malformed user input with clear error messages.
- Improved Modularity: Refactored repetitive code blocks, such as the menu interface, into reusable functions, improving maintainability.
- Enhanced Readability: Renamed variables for better clarity (e.g., preList was changed to prerequisitesList).
- Comprehensive Documentation: Added inline comments to explain complex operations like file parsing and sorting logic, making the code easier to understand and maintain.
- Unit Tests: Developed test cases to verify the reliability of file parsing, sorting, and data display functionalities.

Outcome Coverage

The enhancement addresses the course outcome planned in Module one, specifically:

- Designing and evaluating computing solutions: the enhancements involved evaluating the existing code for areas of improvement and designing solutions by using algorithmic principles and software engineering practices.
- Implementing innovative techniques: by modularizing code, strengthening error handling, and improving documentation, I have attempted to demonstrate innovative problem-solving skills and tools to implement industry-relevant solutions.

Reflection:

Enhancing this artifact taught me several key lessons:

- Importance of Modularity: Refactoring repetitive code into reusable functions improved both the readability and maintainability of the program.
- Error Handling: Strengthening input validation highlighted the value of anticipating user mistakes and ensuring robust application behavior.
- Testing: Writing unit tests gave me a deeper appreciation for the role of testing in producing reliable, professional-quality software.

Challenges:

The primary challenge was ensuring that enhancements did not introduce regressions into the existing functionality. Balancing the need for new features with the requirement to maintain existing behavior required careful testing and iteration. Additionally, improving documentation while ensuring it remained concise yet informative was a challenge I addressed through feedback and iteration.

Conclusion:

This enhancement demonstrates my growth as a computer science professional and my ability to design, develop, and deliver high-quality software solutions. It reflects substantial progress toward achieving proficiency in software design and engineering principles and effectively showcases my skills in the ePortfolio.