# Milestone Two Narrative

## Description of the Artifact

This artifact is a C++ program designed to assist with academic advising at ABCU by managing course data. It allows users to load course information from a CSV file, display a list of courses in alphanumeric order, and retrieve detailed information about a specific course, including its prerequisites. The program was originally created as part of a previous course in the Computer Science program, specifically for a project focused on data management and user interaction. The enhanced version, developed in May 2025 for CS 499, includes improvements to efficiency, error handling, and maintainability.

## Justification for Inclusion in ePortfolio

I selected this artifact for my ePortfolio to showcase my skills in software design and engineering, aligning with the Computer Science program outcomes. This project demonstrates my ability to design and implement a robust solution for managing course data, a practical application relevant to academic environments. The enhancements made to the original program highlight my growth as a software engineer and my ability to apply advanced programming concepts.

Specific components of the artifact that showcase my skills include:

* **Efficient Data Structures**: The integration of a hash table (std::unordered\_map) optimizes course lookups, reducing time complexity from O(n) to O(1) on average, demonstrating proficiency in data structures and algorithms.
* **Robust Error Handling**: Enhanced input validation for course numbers and file inputs, along with detailed error messages, reflects my ability to anticipate and mitigate potential issues, ensuring a user-friendly experience.
* **Code Maintainability**: Comprehensive comments and modular code design improve readability and scalability, aligning with professional software engineering standards for collaborative environments.
* **Input Sanitization**: Whitespace trimming and course number format validation enhance the program's reliability, showcasing attention to detail in user input processing.

The artifact was improved by:

* Adding a hash table for faster course lookups.
* Implementing robust input validation to ensure only valid course numbers (e.g., CSCI101) are processed.
* Enhancing error handling with detailed messages, including line numbers for CSV file errors.
* Adding comprehensive comments to improve code readability.
* Sanitizing user inputs by trimming whitespace to prevent errors from accidental spaces.

These improvements demonstrate my ability to evaluate and enhance existing code, manage design trade-offs, and apply algorithmic principles effectively.

## Alignment with Course Outcomes

In Module One, I planned to meet the following Computer Science program outcomes through this enhancement:

1. **Design and evaluate computing solutions using algorithmic principles and computer science practices** (Outcome 3): By integrating a hash table, I optimized the program's performance, demonstrating an understanding of data structures and algorithmic efficiency.
2. **Demonstrate innovative techniques, skills, and tools in computing practices** (Outcome 4): The enhancements, such as input validation and error handling, reflect professional software engineering practices.
3. **Employ strategies for building collaborative environments** (Outcome 1): Improved documentation ensures the code is maintainable and accessible to other developers.

I successfully met these planned outcomes. The hash table implementation directly addresses Outcome 3 by improving efficiency, while the enhanced error handling and documentation align with Outcomes 4 and 1, respectively. I have no updates to my outcome-coverage plans, as the enhancements fully addressed the intended goals. However, I could further explore Outcome 5 (security mindset) in future iterations by adding data validation to prevent potential exploits, such as buffer overflows or injection attacks, though this was not part of the current scope.

## Reflection on the Enhancement Process

Enhancing this artifact was a valuable learning experience that deepened my understanding of software engineering principles. The process began with a code review to identify inefficiencies, such as the linear search in the original printCourseInfo function, and vulnerabilities, such as lack of input sanitization. Implementing the hash table required understanding the trade-offs between memory usage and performance, reinforcing my knowledge of algorithmic principles. I learned to balance efficiency with resource constraints, choosing a hash table for its average-case O(1) lookup time, which is ideal for a course advising system where quick responses enhance user experience.

Adding input validation and error handling challenged me to anticipate edge cases, such as malformed CSV files or invalid course numbers. This process improved my ability to design robust, user-friendly systems. Writing comprehensive comments honed my skills in creating maintainable code, essential for collaborative development environments. I also learned the importance of iterative development, as testing the enhanced code revealed minor issues (e.g., handling empty inputs) that required further refinement.

Challenges I faced included:

* **Integrating the Hash Table**: Ensuring the hash table (std::unordered\_map) was populated correctly alongside the existing vector required careful synchronization in the loadCoursesFromFile function. I resolved this by updating both data structures simultaneously during file parsing.
* **Input Validation**: Defining a robust course number format (alphanumeric, 5-8 characters) required testing various edge cases to ensure no valid inputs were rejected. I addressed this by implementing a dedicated isValidCourseNumber function.
* **Error Message Clarity**: Providing meaningful error messages, especially for CSV file parsing, was challenging due to the variety of potential errors (e.g., missing fields, invalid prerequisites). I overcame this by including line numbers and specific error descriptions.

Overall, this project reinforced the iterative nature of software development and the importance of balancing functionality, efficiency, and usability. These skills are directly applicable to real-world software engineering, where delivering reliable and efficient solutions is critical.

## Conclusion

This enhanced course advising program is a testament to my growth as a computer science professional. It demonstrates my ability to apply data structures, manage design trade-offs, and produce maintainable, efficient code. By including this artifact in my ePortfolio, I showcase my readiness to tackle complex software engineering challenges and contribute to collaborative, professional environments.