# Description of the Artifact

The artifact is a C++ program titled “ABCU Advising Assistance Program,” originally developed for a CS 300 course project in 2024. It reads course data from a CSV file, stores it in a vector of Course objects, and provides a menu-driven interface to display a sorted course list or detailed information about a specific course, including its prerequisites. The program uses basic data structures (vector) and algorithms (linear search and sorting) to manage and display course data.

# Justification for Inclusion in ePortfolio

I selected this artifact for my ePortfolio because it demonstrates foundational skills in data structures and algorithms, which are critical for computer science professionals. The original program uses a std::vector to store courses and employs linear search and sorting algorithms, providing a solid base for showcasing improvements in these areas. The enhancements made for this milestone significantly improve the artifact’s efficiency and robustness, aligning with the Computer Science program outcome of designing and evaluating computing solutions using algorithmic principles and data structures.

The specific components showcasing my skills include:

* • Transition to std::unordered\_map: Replacing the vector with a hash table reduces the time complexity of course lookups from O(n) to average-case O(1), demonstrat-ing my ability to select appropriate data structures for performance optimization.  
  • Cached Sorted Vector: Maintaining a pre-sorted vector eliminates redundant sort-ing operations, showcasing algorithmic efficiency and trade-off management (mem-ory vs. computation time).  
  • Input Validation with Regular Expressions: Adding format validation for course numbers enhances software reliability, reflecting skills in software engineering de-sign.  
  • Optimized Prerequisite Lookup: Using the hash table for prerequisite title lookups improves efficiency, further demonstrating data structure optimization.

These enhancements improve the artifact by making it more efficient, robust, and scalable, suitable for handling larger datasets while maintaining user-friendly functionality.

# Alignment with Course Outcomes

In Module One, I planned to enhance this artifact to meet the following Computer Science program outcomes:  
• Design and evaluate computing solutions using algorithmic principles and data structures: The original plan included optimizing data access and sorting. The transition to std::unordered\_map and cached sorting directly addresses this out-come by improving lookup and display efficiency.  
• Demonstrate innovative techniques in computing practices: The use of regular ex-pressions for input validation and a hash table for data storage introduces modern C++ features and efficient data structures.  
The enhancements align with the planned outcomes, and no updates to the outcome-coverage plan are needed, as the changes successfully target the intended areas.

# Reflection on the Enhancement Process

Enhancing the artifact taught me several valuable lessons. Implementing the std::unordered\_map required understanding its trade-offs, such as increased memory usage for faster lookups, which deepened my appreciation for balancing performance and resource constraints. Maintaining a separate sorted vector was a strategic choice to avoid modifying the hash table’s structure, teaching me how to manage data consistency across multiple structures. Using std::regex for input validation was a new skill, reinforcing the importance of robust error handling in user-facing applications.

Challenges included:

* • Ensuring Data Consistency: Synchronizing the std::unordered\_map and sorted vector during file loading required careful design to avoid discrepancies.  
  • Regular Expression Design: Crafting a regex pattern that correctly validates course numbers (e.g., “CSCI101”) took experimentation to ensure it was neither too strict nor too lenient.  
  • Performance Considerations: I had to evaluate whether the memory overhead of the hash table was justified for typical use cases, which involved analyzing the expected dataset size.

These challenges were overcome by testing with sample CSV files, debugging edge cases (e.g., invalid course numbers), and consulting C++ documentation for std::unordered\_map and std::regex. The process reinforced my ability to design and evaluate computing so-lutions, aligning with the program’s goals.

# Conclusion

This enhanced artifact showcases my growth in applying data structures and algorithms to solve real-world problems efficiently. By optimizing the original program’s performance and robustness, I have demonstrated skills critical to a computer science professional, making this a valuable addition to my ePortfolio.