**Narrative for Milestone Three – Algorithms and Data Structures**

**Artifact Overview**

The artifact I selected for this milestone is a Course Planner System written in C++, originally created for the CS-300: Data Structures and Algorithms course. The application loads course data from a CSV file, stores the information using appropriate data structures, and allows users to interact with it through a command-line menu. Users can print a list of all courses or view detailed information about a single course, including its prerequisites.

**Justification for Inclusion**

I selected this artifact because it demonstrates the practical application of data structures and sorting algorithms, which are fundamental in computer science. The original implementation used a vector to store courses and sorted them using the STL sort() function with a lambda comparator. For enhancement, I implemented a Merge Sort algorithm to explicitly demonstrate my understanding of algorithmic principles and to allow more control over sorting behavior. Additionally, I added input validation, improved the course search functionality using a map for faster lookups, and made the system more robust.

Key improvements that showcase my skills:

* Replaced sort() with a custom Merge Sort implementation
* Added unordered\_map for efficient course ID lookup
* Improved user input validation and error handling
* Refactored code into modular functions for better readability

**Course Outcomes Achieved**

This artifact helped me demonstrate:

* Outcome 3: Design and evaluation of solutions using algorithmic principles and trade-offs
* Outcome 4: Use of innovative data structures (e.g., map vs vector) to improve performance
* Outcome 2: Improving communication through updated documentation is ongoing

**Reflection on the Enhancement Process**

Enhancing this project helped me go beyond just using built-in STL functions. Writing my own Merge Sort gave me a clearer understanding of time complexity and recursion. Switching to unordered\_map significantly improved lookup speed and validating user input made the application more resilient.

One challenge I faced was debugging the recursive Merge Sort implementation, particularly with pointer boundaries and vector slicing. Testing and visualization helped me confirm that my sorting logic produced the expected results. Through this process, I reinforced the importance of algorithm traceability, modularity, and user-centric design.