**📄 Enhanced Narrative (For Word Document)**

**CS 499 Milestone Three**  
**Enhancement Two: Algorithms and Data Structures**  
**Student: Joshua Emmons**  
**Artifact Title: Student Score Sorter with Merge Sort and Binary Search**

**🔍 Artifact Overview**

The artifact I selected is a Python-based Student Score Sorter that originated in my Data Structures and Algorithms course (CS 300). Initially, the program sorted a list of student names and scores using the merge sort algorithm, enabling fast retrieval through binary search. For this milestone, I revisited the artifact to enhance its functionality by introducing new algorithms and data structures, thereby improving performance and scalability in real-world scenarios.

Enhancements included:

* **Adding a Quick Sort implementation** as an alternative to merge sort, allowing for algorithm performance comparison.
* **Implementing a hash map (dictionary)** via defaultdict to group students by score, enabling constant-time lookups for common queries like “Who scored 88?”
* **Benchmarking** the different algorithms to highlight time complexity trade-offs in practice.
* Refactoring for modularity and adding new utility functions for grouping and alternative sorting.
* Retaining binary search and merge sort to contrast with newer enhancements.

These enhancements go beyond improving readability or performance—they introduce new algorithmic logic and data structures, demonstrating practical problem-solving through advanced CS concepts. This practicality is a testament to the real-world relevance of our academic pursuits.

**💡 Why This Artifact?**

I selected this artifact because it reflects my foundational knowledge in sorting and searching algorithms. But more importantly, it allowed me to explore algorithmic **alternatives** and apply **real-world data structure design**.

Quick Sort is one of the most commonly used sorting algorithms in practice. Including it allowed me to demonstrate my ability to implement recursive algorithms and evaluate their behavior through benchmarking.

The addition of a hash map structure (dictionary) to group students by score directly aligns with modern needs in software performance and design. Grouping and retrieving data efficiently is a skill expected in backend systems, data pipelines, and even frontend filtering systems.

Key skills demonstrated:

* Mastery of **divide-and-conquer** and **recursive algorithms**
* Implementation of **constant-time lookups** using dictionaries
* Understanding of **when and why to use different data structures**
* Confidence in evaluating algorithm performance trade-offs

**🎯 Course Outcome Coverage**

In Module One, I set out to meet these outcomes:

* Design and evaluate computing solutions using algorithmic principles.
* Demonstrate the ability to solve logic problems involving data structures.

I’m confident that this artifact now exceeds those expectations. The inclusion of multiple sorting algorithms and data structures, such as hash maps, provides a much more comprehensive view of how choices impact runtime, scalability, and design complexity. This depth of understanding is a result of rigorous exploration and experimentation.

**🔧 Reflection on the Enhancement Process**

Enhancing this artifact became more than a benchmarking exercise—it became an opportunity to deepen my understanding of how algorithms function under the hood and how different tools serve distinct use cases.

What I learned:

* **Quick Sort** reinforces how different recursive strategies can affect time and space complexity.
* **Hash Maps** enable simple yet powerful logic to answer real-world questions fast.
* Writing clean, reusable, and modular functions not only improves testability but also speeds up debugging and iteration.

Challenges:

* Ensuring that performance comparisons between sorting algorithms were fair and consistent.
* Designing meaningful ways to use hash maps without bloating the original scope.
* Avoiding redundancy in code while expanding logic across multiple techniques.

This enhancement turned into a capstone moment for me—a bridge from "knowing" algorithms to **applying** them strategically.

**🚀 Final Thoughts**

This project now demonstrates not just that I know how to use algorithms like merge sort and binary search, but that I can **adapt**, **expand**, and **evaluate** alternative approaches as needed. It demonstrates that I can select and implement the appropriate data structure for the problem, and justify that choice based on performance, readability, and practical application.

I’m proud to feature this enhanced artifact in my ePortfolio. It doesn’t just represent code—it describes how I approach problems, evolve solutions, and keep striving for better. This sense of accomplishment reflects the effort and dedication I've put into this project.