Darius Stewart

Southern New Hampshire University

CS-499 Computer Science

Professor Kalinowski

6/1/2025

Milestone Three Narrative – Algorithms and Data Structures

**Artifact Description:**

The artifact I chose for this milestone is a Spring Boot–based Java application originally developed in CS-305. It serves a simple RESTful endpoint that takes an input string and returns its SHA-256 hash. The application demonstrates secure computing principles such as HTTPS communication, modularity, and message digest integrity. It was created to highlight secure server design, but I’ve now enhanced it further to reflect core skills in algorithms and data structures.

**Why I Included This Artifact:**

I selected this artifact for the "Algorithms and Data Structures" category because it offers a strong foundation for demonstrating real-world applications of efficient data handling. Specifically, I incorporated a ConcurrentHashMap to cache previously computed SHA-256 hashes, which significantly improves the algorithm’s efficiency by avoiding redundant computation. This enhancement allowed me to show not only my understanding of hashing algorithms but also my ability to optimize performance through appropriate data structures.

**Enhancement Summary and Outcome Mapping:**

This enhancement maps directly to the following Computer Science program outcome:

"Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution."

To meet this outcome, I integrated an in-memory cache using ConcurrentHashMap. This thread-safe data structure ensures that concurrent requests to hash identical inputs will benefit from previously computed values, reducing CPU usage and improving performance. The use of computeIfAbsent also demonstrates my familiarity with lambda-based functional techniques in modern Java, which simplifies the logic and avoids race conditions.

This change builds on prior enhancements made in Week 3, which focused on modularization and secure design. Together, these show layered improvements in both design and algorithmic thinking.

**Reflection and Learning:**

Implementing the caching layer made me think more deeply about efficiency, concurrency, and practical trade-offs in system design. I learned how to integrate thread-safe collections in a REST API context and how to write cleaner, more concise logic with Java’s functional interfaces. It also deepened my understanding of how algorithms and data structures aren’t just theoretical topics—they drive performance and scalability in production systems.

One challenge I faced was deciding where the responsibility for caching should lie: in the controller layer or deeper in a service class. For this small application, placing it in the controller was justified, but I now recognize the value of layering responsibilities more cleanly in larger systems.