Nicholas Wyrwas

CS-499-10452-M01

Mrs. Philomena Ogoh

10/05/2025

4-2 Milestone Three: Enhancement Two Algorithms and Data Structure

The artifact I chose for this milestone is the same Client/Server Development Project that I initially created for CS-340. This Python Flask server application backed by a MongoDB database to perform CRUD operations on an animal shelter database. The initial version met all requirements but left opportunity for optimization by applying more effective algorithms and data structures. In the current iteration, I am seeking to improve performance and scalability by fine-tuning the algorithmic logic of the project and adding a caching layer to store data in memory. This applies the algorithmic thinking and data structure design criteria from the course outcomes of the computer science program.

I chose this artifact because it provides an opportunity to take a basic but functional database-backed program and apply algorithmic thought. The initial version of this program scanned the entire database for all reads, which resulted in increasing latency over multiple repeated reads of the same data. To optimize this, I added a Least Recently Used in-memory cache for common read queries by leveraging Python functools.lru\_cache to store query results locally in-process with a capped cache size. This optimization decreased the average read latency and provided a lesson in how caching strategies can speed up responses without altering original data. In addition, I optimized read queries to rely on MongoDB indices of common and high frequency filter terms such as animal\_type, breed, and age fields, reducing time complexity for common read operations for these indexed fields. Both of these optimizations are data structure and algorithm improvements made in real-world software systems to improve efficiency, throughput, and maintainability.

By this improvement I have met many of the associated course outcomes. I diagnosed the performance bottleneck and used the algorithm design knowledge to identify and integrate a more effective solution. I have deepened my understanding of data structures by employing two such data structures in the form of caching and indices, which are both built on top of hash maps and priority ordering data structures. I also identified tradeoffs between time and memory usage, as caching allows for faster responses at the cost of increased memory consumption. Overall this has strengthened my sense of informed decision making aligned with industry practices.

In order to perform this work I had to think about both the underlying technical details and about how to execute my own work. The caching layer required validation steps to ensure cache results stay in sync with database state, such as cache invalidation on updates and deletes, and I performed small benchmarks to verify improvement. I had some blockers in the form of other work and personal obligations that pushed out some hands-on work earlier in the term than was originally planned; this was remedied by scheduling two dedicated 90-minute work blocks on my calendar each week to ensure there was dedicated and predictable time to focus on completing the enhancements and writing the documentation.

All in all, this milestone has brought me further along in my understanding of how algorithms and data structures apply to a concrete client/server system. By integrating caching, indexing, and validating these operations in a disciplined way, I have improved its performance and deepened this mindset. The experience has also reinforced good habits around testing, documentation, and time-blocking, which will be of value in future professional software work.