Milestone Three: Algorithms and Data Structures Enhancement  
Michael Ansah-Akrofi  
CS 499 – Computer Science Capstone

Artifact Description

The selected artifact is a Python-based CRUD application called animal\_shelter.py, originally developed in a previous course as part of the Grazioso Salvare project. The program connects to a MongoDB database to manage animal records from the Austin Animal Center. The original version of the code included basic methods to create, read, update, and delete records. It utilized the pymongo library to interface with MongoDB.

The initial project was created in an earlier module, and while functional, it lacked advanced logic, error handling, and algorithmic efficiency in how data was queried and processed. For Milestone Three, I focused on enhancing the artifact to better showcase my understanding of algorithmic problem-solving and efficient data structure usage.

Justification for Inclusion

I selected this artifact for the Algorithms and Data Structures category because it allowed me to apply advanced data manipulation techniques and optimization strategies to real-world operations on a NoSQL database. Specifically, I incorporated sorting algorithms, optimized search queries using MongoDB's aggregation framework, and implemented filtering logic with condition chaining.

The enhanced version of the artifact now demonstrates:

* Use of aggregation pipelines to process data more efficiently
* Structured query optimization to reduce runtime complexity
* Implementation of condition-based filtering for specific use cases (e.g., identifying adoptable animals by breed, age, or training status)
* Encapsulation of reusable logic through well-structured helper functions

These enhancements reflect a deeper understanding of both algorithm design and practical data handling using appropriate structures (lists, dictionaries, conditional logic, etc.).

Course Outcomes Addressed

This enhancement directly supports the course outcome:

“Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices.”

In Module One, I planned to improve the algorithmic logic of my artifact by integrating sorting and aggregation to improve performance and usability. I’ve met that goal and exceeded it by also adding more robust filtering and error handling to ensure the application handles diverse edge cases.

Reflection on the Enhancement Process

During the enhancement process, I gained practical experience refining algorithmic logic for real-world database applications. I learned to balance readability with performance and modularity—refactoring long blocks of code into smaller, reusable functions that follow the single-responsibility principle. I also became more comfortable with MongoDB’s aggregation framework and how to design pipelines that perform operations like filtering, grouping, and sorting—all without needing to pull large datasets into memory in Python.

One challenge I faced was handling cases where the aggregation query returned no data. Initially, this would cause the program to crash or return unclear results. I addressed this by adding condition checks and default responses, improving the robustness and reliability of the code.

Conclusion

This enhancement represents significant progress in my competency with algorithm design and data structures. The final artifact not only works more efficiently but is also more readable, scalable, and professional—ready to be included in my ePortfolio. It stands as evidence of my ability to apply theoretical computer science knowledge to practical problems, aligning with industry expectations for data-driven application development.