CS-499 Milestone Four  
Enhancement Three: Databases

Shawn Henly

06/09/25

### **Brief Description of the Artifact**

The selected artifact for this milestone is the animal\_shelter.py module originally developed in CS-340: Advanced Programming Concepts. This Python class was created for Grazioso Salvare, a fictional dog rescue organization, and it enables basic CRUD operations for interacting with a MongoDB database. The original version allowed the insertion, retrieval, updating, and deletion of animal records from the MongoDB animals collection. However, it featured hardcoded credentials, lacked error handling, did not validate input, and did not use logging or indexing. While functionally correct, its structure was brittle and not aligned with secure or scalable development practices.

### **Justification for Inclusion in the ePortfolio**

I selected this artifact because it represents my progression from writing basic data access logic to implementing a secure, maintainable, and performance-aware database module. The original version was created with minimal safeguards and was difficult to adapt for real-world use. In enhancing this artifact, I was able to demonstrate growth in secure software development, defensive programming, and data-handling best practices.

This artifact showcases my skills in back-end system integration, error handling, and logging—areas essential to modern application development. By refactoring the class to better separate concerns, integrating Python’s built-in logging framework, and supporting credential management through environment variables, I transformed the script from a prototype to a component. This work not only makes the module safer and more scalable but also communicates my readiness to develop reliable data services in industry environments.

### **Key Components of the Enhancement**

* **Credential Management with Environment Variables**: I used Python’s os module to support secure retrieval of MongoDB credentials and host information from environment variables while preserving default values for testing.
* **Logging Integration**: Replacing untracked operations with Python’s logging module, I added visibility into each method’s behavior. This makes debugging and auditability far easier—important for real-world maintainability.
* **Try-Except Error Handling**: Each CRUD method is now wrapped in try-except blocks to prevent crashes and capture database-related exceptions from pymongo.errors.
* **Performance Optimization**: Indexes were added for commonly queried fields such as breed, location, and adoption\_status. These improve database read performance, especially on large datasets.
* **Validation and Fallbacks**: Default values and input type checks ensure methods handle bad input gracefully. For instance, if an invalid query object is passed to read(), the method logs a warning and returns an empty list.

### **Course Outcomes Achieved**

This enhancement supports several CS-499 outcomes. It aligns with **Outcome 3**, which focuses on designing computing solutions using appropriate principles and managing design trade-offs. I restructured the data access code to separate responsibilities, optimize queries, and increase robustness. These decisions reflect a thoughtful balance between complexity, performance, and security.

It also demonstrates progress toward **Outcome 4**, which involves the use of well-founded and innovative tools. The integration of logging and error handling tools, along with the introduction of indexes and secure credential handling, highlights my use of professional-grade techniques to support reliability and observability.

Finally, the work addresses **Outcome 5**, which calls for a security-oriented mindset. I eliminated hardcoded credentials and protected against common failure points, improving the overall integrity and security of the application.

Because these enhancements were fully aligned with the plans and goals I defined in Module One, no changes to my outcome coverage plan are needed at this time.

### **Reflection on the Enhancement Process**

Enhancing the animal\_shelter.py module gave me direct experience with improving production readiness of data access code. Initially, the class served its educational purpose, but it lacked the structure and defensive design required for maintainable software. Moving the credentials into environment variables was a small but meaningful step toward secure coding—one I hadn’t prioritized in earlier courses.

Working with exception handling in PyMongo and logging revealed the importance of designing for failure. I also learned how indexes can make a substantial difference in performance for search-heavy applications.

The most valuable takeaway was understanding how seemingly “working” code can still fail when scaled, reused, or deployed. This experience taught me to look beyond functionality and begin thinking about long-term safety, extensibility, and supportability.