CS 499 Computer Science Capstone

5-2 Milestone Four: Enhancement Three: Databases

Professor Dr. Pravin Bhandari

Mubeen Ahmed Khan

mubeenahmed.khan@snhu.edu

Southern New Hampshire University

6th October, 2025

Enhancement Three: Databases

Artifact: animal_shelter.py (MongoDB CRUD module)

Overview of the Artifact

The artifact I selected for this milestone is the same *Animal Shelter CRUD Module* that I originally developed in CS-340: Client/Server Development. It is a Python-based MongoDB interface that powers a Jupyter Notebook dashboard, providing the ability to create, read, update, and delete (CRUD) animal records while visualizing them through interactive charts and geolocation maps.

For this enhancement, the focus was on strengthening the database layer — improving data integrity, efficiency, and scalability. Specifically, the enhancements included adding JSON schema validation within MongoDB, implementing optimized aggregation pipelines, creating indexes for faster query performance, and improving database-level security and error handling.

Why I Included This Artifact

I chose this artifact because it demonstrates practical, real-world database management principles that align directly with both data engineering and software design outcomes. Modern systems depend heavily on clean, validated, and efficiently queried data. Enhancing the *animal_shelter.py* module with schema enforcement, indexes, and server-side analytics reflects my growth from a software developer into a more database-conscious engineer.

The enhancements I made showcase several important professional skills:

- **Schema Validation:** I introduced a MongoDB $jsonSchema validator that enforces structural and data-type integrity at the database level, ensuring malformed or inconsistent data cannot be inserted.

- **Server-Side Aggregation:** I implemented $match, $group, $sort, and $limit aggregation pipelines to calculate top animal breeds and related summaries directly in MongoDB.
- **Indexing for Performance:** I created compound and geospatial indexes on key fields like species, breed, and location to improve query and geolocation performance.
- **Enhanced Robustness:** I incorporated better logging, validation, and optional environment-driven configuration for flexibility and maintainability.

Course Outcomes Addressed

In Module One, I planned to meet the outcome of demonstrating the ability to use innovative techniques, tools, and practices in computing to implement database solutions that deliver value and ensure data security. Through this enhancement, I met that outcome by:

- Designing and implementing MongoDB schema validation rules for consistent data structure.
- Using aggregation pipelines to push computational logic to the database, reducing data transfer and improving performance.
- Building indexes to enhance query speed and geospatial lookups.
- Incorporating environmental configuration and error handling for deployment-ready reliability.

These improvements also strengthen outcomes in **software engineering/design** and **security**, as they improve the structural design of the module while mitigating risks of unsafe writes or inefficient queries.

Reflection on the Enhancement Process

Working on this enhancement deepened my understanding of how database-level improvements translate into application-level benefits. I learned that enforcing structure and validation close to the data source prevents many downstream errors, saving debugging time and ensuring consistent analytics.

For example:

- Adding the `$jsonSchema` validator required balancing strictness with flexibility. I had to allow optional fields while ensuring numeric ranges and valid geographic coordinates.
- Designing aggregation pipelines taught me the importance of letting MongoDB handle large-scale computations rather than transferring unnecessary data to Python.
- Creating indexes revealed how much performance can be gained through thoughtful data organization—especially when dealing with search filters or map queries.

The process also reinforced professional habits such as iterative testing, schema versioning, and defensive programming, which are crucial when evolving a live data system.

Challenges and Resolutions

Challenge	**Resolution**
MongoDB schema validation (`collMod`) required admin privileges and could fail on restricted environments.	Implemented an optional .env flag (`MONGO_APPLY_VALIDATOR`) to toggle validator enforcement safely.

Challenge	Resolution
Performance optimization risked breaking existing functionality.	Ensured backward compatibility by maintaining existing method signatures and behaviors while refactoring internals.
Aggregation pipelines needed to handle diverse data while returning results suitable for the Dash UI.	Used $match, $group, and $sort with dynamic filters and ensured the output format matched visualization needs.
Index creation had to be idempotent to avoid errors on restarts.	Encapsulated index setup in `_ensure_indexes()`, which checks and safely recreates indexes when needed.
Unit tests needed to confirm both aggregation correctness and validator safety.	Added new integration tests (`test_aggregation.py`) that verify aggregation counts and validator behavior without corrupting production data.

How the Artifact Was Improved (Concrete Changes)

- Implemented $jsonSchema validation with optional activation through .env (`MONGO_APPLY_VALIDATOR=1`).
- Added `_ensure_indexes()` method to create standard and geospatial indexes automatically at startup.
- Developed a `top_breeds()` method that uses MongoDB's aggregation pipeline for efficient analytics.
- Added environment-driven configuration for database URI, name, and collection to support portability and deployment flexibility.

- Enhanced logging for all CRUD operations and schema enforcement actions.
- Added an integration test suite (test_aggregation.py) to verify database aggregation and validation behavior.
- Updated the dashboard's graph callbacks to use server-side aggregation, improving performance and scalability.

Possible Indicators of Success

This enhancement aligns directly with the indicators of success for **database design and management**:

- I demonstrated the ability to use database indexing, aggregation, and validation techniques to improve system performance and data integrity.
- I implemented computing practices and tools that deliver measurable improvements in efficiency and security.
- I balanced design trade-offs between validation strictness and operational flexibility, showing professional-level database administration insight.
- I supported my implementation with meaningful testing, documentation, and consistent coding style across modules and notebooks.

Summary

Enhancing the *Animal Shelter CRUD Module* at the database layer has transformed it into a more complete, production-ready artifact. It now combines clean schema design, efficient query performance, and flexible configuration with improved fault tolerance. This enhancement illustrates my ability to merge database theory with practical application — designing, implementing, and validating solutions that ensure both data integrity and system scalability.