CS 499 Computer Science Capstone

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

Professor Dr. Pravin Bhandari

Mubeen Ahmed Khan

mubeenahmed.khan@snhu.edu

Southern New Hampshire University

28th September, 2025

Enhancement Two: Algorithms and Data Structure

Artifact: animal_shelter.py (MongoDB CRUD module)

Overview of the Artifact:

The artifact I selected is my **Animal Shelter CRUD Module**, created in **CS-340: Client/Server Development**. It is a Python module (animal_shelter.py) paired with a Jupyter Notebook dashboard that interacts with a MongoDB database. The module provides Create, Read, Update, and Delete (CRUD) functionality, and the notebook demonstrates how the data can be visualized in tables, graphs, and maps.

Why I Included This Artifact

I chose this artifact because it demonstrates clear application of **algorithms and data structures** in solving a real-world problem—efficiently managing and querying large datasets. The enhancements I made showcase my ability to optimize algorithms, structure data for performance, and build safeguards into the system. Specifically, I added:

- **Data Cleaning Algorithms**: Introduced a helper (coerce_lat_long) to coerce latitude and longitude values into numeric types, preventing invalid values from breaking the map visualizations.
- **Caching Mechanisms**: Implemented both a **data-layer cache** in the CRUD module and a **UI-layer cache** in the notebook to minimize redundant queries and re-computations.
- **Validation Algorithms**: Added stricter filter and input validation to block unsafe queries, enforce correct types, and handle missing or malformed values gracefully.

- **Refined Query Operations**: Optimized the `read()` method to support projections, safe filters, and pagination—showcasing efficiency in retrieving structured data.

These improvements not only highlight my technical ability but also demonstrate my awareness of **performance trade-offs, data integrity, and user experience**.

Course Outcomes Addressed

In Module One, I planned to meet the outcome of **designing and evaluating computing solutions using algorithmic principles and appropriate data structures**. I achieved this by:

- Designing caching algorithms that reduce database load.
- Validating inputs and coercing types to maintain structural integrity.
- Optimizing queries to balance efficiency, clarity, and security.

Additionally, the enhancements support outcomes in **software engineering/design** and **security**, since the code was refactored for maintainability and explicitly guards against unsafe query patterns.

Reflection on the Enhancement Process

Enhancing this artifact taught me valuable lessons about balancing **performance, maintainability, and robustness**. For example:

- Implementing caching highlighted how small algorithmic improvements can dramatically improve user-facing performance.
- Adding validation and `NaN-handling` for `lat/long` values underscored the importance of **defensive programming** in data-intensive applications.

- Writing unit tests (both with mocked collections and integration smoke tests) reinforced the need to verify that algorithms behave predictably across both expected and edge cases.

The most challenging part was ensuring that the improvements did not break the existing interface. Keeping the method signatures the same required careful design, but it also demonstrated my ability to enhance functionality **without disrupting downstream dependencies**—a critical skill in professional software engineering.

Challenges and Resolutions

- **Challenge**: Cached queries needed to remain consistent after data modifications.
  - **Resolution**: Added automatic cache invalidation in create, update, and delete methods.
- **Challenge**: Latitude and longitude values were sometimes stored as strings or invalid values, breaking map displays.
  - **Resolution**: Created the coerce_lat_long function to convert to numeric types and drop invalid entries before visualization.
- **Challenge**: MongoDB filters could allow unsafe or unintended operators.
  - **Resolution**: Implemented _validate_filter to block unsafe $ operators and allow only a safe subset of update operations.
- **Challenge**: Needed to improve performance without changing how external code interacted with the module.
  - **Resolution**: Maintained the original class structure and method signatures, but refactored internals for optimization and clarity.

- **Challenge**: Tests had to demonstrate caching speedup reliably without flakiness.
    - **Resolution**: Used mocked delays (time.sleep) in unit tests to show measurable performance improvements between uncached and cached reads.

How the Artifact Was Improved (Concrete Changes)

- Added **in-memory caching** in the CRUD module (animal_shelter.py) to reduce redundant database reads.
- Enhanced the **Jupyter Notebook** with a UI-level cache to store filtered results and precomputed chart data.
- Implemented a **data cleaning helper** (coerce_lat_long) and integrated it into the notebook at three key points: initial load, table filter updates, and map updates.
- Strengthened **validation** by enforcing safer query filters, restricting update operators, and handling NaN values in geographic data.
- Refactored code for **readability and maintainability**, while keeping public method signatures unchanged.
- Expanded **unit tests** to cover caching behavior, invalid inputs, error handling, and timing improvements.

Possible Indicators of Success

This enhancement aligns directly with the indicators of success for algorithms and data structures:

- I demonstrated the ability to use **innovative techniques** (such as in-memory caching and NaN-safe validation) to implement design solutions that accomplish project goals.

- I programmed solutions to solve **logic problems** involving algorithms and data structures, particularly in query optimization and cache management.
- I addressed potential **design, logical, and structural flaws related to security** by validating inputs, blocking unsafe query operators, and guarding against malformed data.
- I clearly articulated my **ideas and accomplishments** by producing a working product with clean code, inline documentation, and unit tests that show the value of my enhancements.