Jason Nasr

April 3, 2025

CS 499

Milestone 4 Narrative

The artifact I selected is a **web-based dashboard and backend system** that connects to a MongoDB database to perform CRUD operations (Create, Read, Update, Delete) on animal shelter records. The backend component, implemented in Python, interacts directly with a MongoDB instance using a custom AnimalShelter class. The frontend, built with Dash (a Python framework), displays filtered records through tables, charts, and an interactive map.

I chose this artifact for my ePortfolio because it demonstrates a **full-stack solution** involving both **database design/management** and **interactive data visualization**. It represents my ability to integrate backend logic with frontend interfaces using real-world tools and techniques that are commonly used in modern software development.

### Skills and Components Showcased

* **Database Integration & Management:** Connection to a real MongoDB database, with secure credential handling, query filtering, and schema-aware data validation.
* **CRUD Implementation:** Modular, well-documented code using modern MongoDB methods (insert\_one, update\_many, etc.), with input validation and exception handling.
* **Dash Visualization:** Built a responsive and user-friendly dashboard that includes charts (Pie chart using Plotly), an interactive table, and a location-based map with leaflet integration.
* **Error Handling & Defensive Programming:** Incorporated exception handling and user input validation throughout the code to ensure robustness.
* **UI/UX Awareness:** Improved data readability and interactivity with dropdown filters, styled tables, and intuitive data displays.

### **Artifact Enhancements**

The artifact was significantly improved in the following ways:

* Replaced the Apporto virtual MongoDB with a local MongoDB configuration, making the project more portable and realistic.
* Rewrote the CRUD module to include input validation, modern method usage, and robust error handling.
* Enhanced the frontend layout with cleaner UI structure, updated interactivity logic, and safe defaults for all user inputs.
* Added inline comments and full docstrings for clarity, reusability, and professional readability

**Database Enhancments**

* Replaced the default connection to Apporto’s virtual server with support for **local MongoDB** (localhost:27017).
* Parameters for host, port, database, and collection are now customizable via the constructor.

#### **2. Robust Error Handling**

* Wrapped all MongoDB operations (insert\_one, find, update\_many, delete\_many) in try-except blocks.
* Meaningful error messages are printed when operations fail, aiding debugging and preventing crashes.

#### **3. Input Validation**

* Added strict type and empty-checks for input arguments (data, criteria, initial, change, remove) in all methods.
* Prevents the database from receiving malformed queries or null operations.

#### **4. Modern MongoDB Methods**

* Replaced deprecated methods like .insert() with **.insert\_one()**.
* Uses **.update\_many()** and **.delete\_many()** with result metadata (raw\_result) returned for feedback.

#### **5. Read Method Compatibility**

* Automatically removes the \_id field in read() responses using projection ({"\_id": False}), ensuring compatibility with frontend components like Dash tables.

#### **6. Clear Inline Comments and Docstrings**

* Every method is now fully documented with a docstring explaining:
  + What it does
  + What arguments it takes
  + What it returns
* Comments explain why certain decisions (like dropping \_id) are made.

In Module One, I aimed to meet the following course outcomes:

* **Design and develop data-driven applications** that are maintainable and scalable.
* **Utilize industry-standard tools** and practices to solve real-world software problems.
* **Demonstrate full-stack development** skills with backend and frontend integration.

With this enhancement, I successfully met all the planned outcomes. No updates are necessary to my outcome-coverage plan, as the revised artifact covers each competency area effectively.

The enhancement process was both technically enriching and personally rewarding. I learned:

* How to structure more **modular and maintainable code**, especially in terms of separating logic for connection, data operations, and UI.
* Best practices for **MongoDB interaction in Python**, including modern syntax and performance-aware queries.
* How to **create a seamless flow of data** between a backend and a user-facing dashboard.