**5-2 Milestone Four: Enhancement Three: Databases**

Name: Ami Akagi

Date: August 3rd, 2025

Milestone Four Narrative – Database Enhancement

**Overview of the Artifact**

This artifact is a web application that manages animal shelter data. Data is stored in MongoDB, which captures key elements about each rescued dog's color, age, breed and the type of rescue it was picked up from.

Main features include:

* A RESTful API for data create, update and delete
* Importing and validating data from large-scale CSV files
* Type of rescue and breed Classification; normalization
* Storage of geographic location data

**Why This Artifact**

I selected this artifact because it helped me develop fundamental skills in working with databases.

* Learn how to Design your NoSQL database with MongoDB and MongoEngine
* Django REST Framework for Building RESTful APIs
* Data from large scale CSV files > importing and validating
* Database connection errors and exception for duplicate key entries

Key improvements I made include:

* Addressed case sensitivity issues in database naming conventions
* Proper serialization of ReferenceFields
* Coming up with Smart Caching strategies to improve performance

**Alignment With Course Outcomes**

Through this enhancement, I was able to apply and demonstrate alignment with several Computer Science program outcomes. I was able to do this, by creating a database schema (to achieve Outcome 2), writing a secure backend API (Outcome 3), and integrating multiple technologies together as a fully cohesive application in order to complete the CR app (Outcome 4).

**Development Process and Takeaways**

I first started refining this artifact and got a mix of things that I was not expecting. I thought it would be fairly easy, but as I got into it there were a lot of little things to change.

One issue, for instance, is how MongoDB deals with the conversion of collection names to lowercase. This of course caused some difficulties at first, particularly when needing to work across cloud providers. Another task was to implement a serialisation which could handle circular reference objects like a dog having breed objects linked. I had never seen this before but it took me time. That part turned out to be more complex than I initially expected.

When importing the big dog set of data from CSV files. The import process also presented challenges due to formatting errors in some records. A few of the records were bad, and I had to add some validation that would not allow duplicates or stop the import if a format was wrong. After I added in some logging and made sure that we caught duplicates by animal\_id, this was much better.

With that being said, it was actually working through these things that made me realize how messy data and real-world edge cases are handled in actual applications. Great practice for debugging, writing cleaner logic, or just making something that doesn't fall apart when handling unexpected or poorly structured input.

**Technical Details**

Database Design:

* dogs, breeds and rescue\_types are the three MongoDB collections
* Indexed for efficient lookup
* Structured for scalability and flexibility

API Implementation:

* RESTful endpoints with full CRUD support
* Status codes and exception handling
* Collections with references between them managed by IDs

Data Processing:

* Script to import CSV with validation logic
* Duplicate-checking using unique animal\_id
* MongoDB connection based upon environment to make it more secure.

**Conclusion**

I learned a great deal through this project, including how to design a database, build an API, and manage large datasets effectively. And then, I got a workable, scalable backend to be integrated within an entire system. These are essential foundational skills for aspiring full-stack and backend engineers in future software development projects.

**Technology Stack:**

* MongoDB (Database)
* Django (Web Framework)
* Django REST Framework (API Development)
* MongoEngine (ODM)
* Python

**Project Files:**

* backend/api/models.py – database models
* backend/api/views/ – API views
* backend/load\_data.py – CSV import logic
* backend/requirements.txt – dependency list