## 

## 

## 

## 

## 

## 

## 

## **3-2 Milestone Two: Enhancement One: Software Design and Engineering**

Name: Ami Akagi

Date: July 20th, 2025

## **Enhancement Summary**

### **1. Moved Sensitive Credentials to .env**

To improve security and follow best practices, I removed all hardcoded database credentials from the source code. Instead, I created a. env file and put the MongoDB connection details (host, port, username, password, database, collection) in there.

In both animal\_shelter. py and ProjectTwoDashboard.ipynb, I was loading these variables using the .tenv package:

**ProjectTwoDashboard.ipynb**

# Load environment variables from the .env file

load\_dotenv()

# Avoids hardcoding sensitive information

username = os.getenv("MONGO\_USERNAME")

password = os.getenv("MONGO\_PASSWORD")

**animal\_shelter.py**

load\_dotenv()

HOST = os.getenv("MONGO\_HOST")

PORT = ***int***(os.getenv("MONGO\_PORT"))

DB = os.getenv("MONGO\_DB")

COL = os.getenv("MONGO\_COL")

This prevents sensitive information from being exposed in the codebase and allows safer deployment and version control.

**2. Improved Filtering Logic for Rescue Types**

Originally, it was only possible to query the dogs one rescue type at a time (Water, Mountain, Disaster). I added logic to get filtered data by using multi-select options instead of comparing hardcoded values, instead I used $in query operator. The reason behind this is to keep the dashboard more flexible and usable.

breed\_map = {

"water": ["Labrador Retriever Mix", "Chesapeake Bay Retriever", "Newfoundland"],

"mountain": ["German Shepherd", "Alaskan Malamute", "Old English Sheepdog", "Siberian Husky", "Rottweiler"],

"disaster": ["Doberman Pinscher", "German Shepherd", "Golden Retriever", "Bloodhound", "Rottweiler"]

}

if 'all' in selected\_types or not selected\_types:

selected\_types = ['water', 'mountain', 'disaster']

selected\_breeds = []

for rescue\_type in selected\_types:

selected\_breeds.extend(breed\_map.get(rescue\_type, []))

query = {

"breed": {"$in": selected\_breeds},

"age\_upon\_outcome\_in\_weeks": {"$lte": 104}

}

**3. "All" Option with Exclusive Behavior**

To improve the user experience, I implemented logic to manage the "All" option in the checklist:

* Choose “All” and other types will clear. The checkbox “All” is removed when the user chooses another type along with it.  
  In case when all individual types are chosen, the app changed them all to "All" for simplicity.
* When nothing is selected "All" by default is the fallback.

if not selected:

return ['all']

# If 'all' is selected with other options, remove 'all'

if 'all' in selected and len(selected) > 1:

selected.remove('all')

# If only one value is selected and it's 'all', keep it

if selected == ['all']:

return ['all']

# If all 3 rescue types are selected, convert it to 'all'

all\_rescues = {'water', 'mountain', 'disaster'}

if ***set***(selected) == all\_rescues:

return ['all']

return selected

This will make it more intuitive and consistent what filtering does, and prevent confusion for the user evident now.