Grazioso Salvare – Rescue Dog Finder Dashboard

A Dash-based web application that connects to a MongoDB database (Austin Animal Center “Outcomes” data set) to help Grazioso Salvare identify dogs suitable for specific search-and-rescue roles. The app supports secure DB access, interactive filters, a live data table, a geolocation map, and a summary chart.

**Author:** Ivan Gonzalez  
 **Org:** Global Rain (course project)

## **About the Project**

Grazioso Salvare needs to quickly locate dogs that match rescue profiles (e.g., water rescue, mountain/wilderness rescue, and disaster/individual tracking). This dashboard reads from a MongoDB collection of shelter outcomes and lets users filter, browse, and visualize candidates.

## **Features**

* **Secure DB connection** using aacuser credentials (created in Module 3).
* **Interactive filters** (radio buttons) for:
  + Water Rescue
  + Mountain/Wilderness Rescue
  + Disaster/Individual Tracking
  + Reset (All Dogs)
* **Data table** (sortable, paged, single-row selectable).
* **Geolocation map** (Leaflet) that pans to the selected row’s coordinates.
* **Second chart** (breed distribution) that updates with the filter.
* **Branding**: Grazioso Salvare logo and a unique identifier (“Ivan Gonzalez”).

## **Architecture**

* **Model**: MongoDB (aac.animals) accessed via a reusable Python **CRUD module**.
* **View**: Dash UI components — filters, data table, charts/map.
* **Controller**: Dash callbacks that call the CRUD functions and update views.

## **Getting Started**

Run inside Codio / JupyterLab.

1. **Verify MongoDB & Data**
   1. mongoimport has already loaded datasets/aac\_shelter\_outcomes.csv into DB aac, collection animals.
   2. Ensure user aacuser exists in admin with readWrite on aac.
2. **Set credentials in CRUD**
   1. CRUD\_Python\_Module.py defaults:  
       user="aacuser", passwd="D@rkLumoo28", host="127.0.0.1", port=27017, db="aac", col="animals".
3. **Run the dashboards**
   1. **ModuleFiveAssignment.ipynb** (authentication demo).
   2. **ModuleSixMilestone.ipynb** (table + map).
   3. **ProjectTwoDashboard.ipynb** (full dashboard).

## **Installation**

Within the Codio/Jupyter environment:

pip install pymongo jupyter-dash dash dash-leaflet plotly pandas

Everything else (NumPy/Matplotlib) is typically preinstalled.

## **Usage**

### **Start the Project Two dashboard**

1. Open **ProjectTwoDashboard.ipynb** and run all cells.
2. The app renders inline (if port 8050 is busy the code tries a fallback).

### **Interactions**

* Choose a **rescue filter** — table, map, and chart update.
* Click a **row** in the table to move the map marker to that animal’s location.
* Use **Reset** to see all dogs again.

## **Filtering Logic & Queries**

We apply filters based on the “Rescue Type and Preferred Dog Breeds” table.

We use age\_upon\_outcome\_in\_weeks to simplify age ranges.

### **1) Water Rescue**

* **Breeds**: Labrador Retriever Mix, Chesapeake Bay Retriever, Newfoundland
* **Sex**: Intact Female
* **Age**: 26–156 weeks
* **Type**: Dog

{  
 "animal\_type": "Dog",  
 "sex\_upon\_outcome": {"$regex": "^Intact Female"},  
 "age\_upon\_outcome\_in\_weeks": {"$gte": 26, "$lte": 156},  
 "breed": {"$in": ["Labrador Retriever Mix", "Chesapeake Bay Retriever", "Newfoundland"]}  
}

### **2) Mountain or Wilderness**

* **Breeds**: German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, Rottweiler
* **Sex**: Intact Male
* **Age**: 26–156 weeks
* **Type**: Dog

{  
 "animal\_type": "Dog",  
 "sex\_upon\_outcome": {"$regex": "^Intact Male"},  
 "age\_upon\_outcome\_in\_weeks": {"$gte": 26, "$lte": 156},  
 "breed": {"$in": ["German Shepherd", "Alaskan Malamute", "Old English Sheepdog",  
 "Siberian Husky", "Rottweiler"]}  
}

### **3) Disaster or Individual Tracking**

* **Breeds**: Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, Rottweiler
* **Sex**: Intact Male
* **Age**: 20–300 weeks
* **Type**: Dog

{  
 "animal\_type": "Dog",  
 "sex\_upon\_outcome": {"$regex": "^Intact Male"},  
 "age\_upon\_outcome\_in\_weeks": {"$gte": 20, "$lte": 300},  
 "breed": {"$in": ["Doberman Pinscher", "German Shepherd", "Golden Retriever",  
 "Bloodhound", "Rottweiler"]}  
}

### **4) Reset**

{"animal\_type": "Dog"}

The controller passes these queries into AnimalShelter.read(query, projection).







