Austin Johnson

Mr. Ling

CS 340

08/24/25

# **Grazioso Salvare Dashboard (CS‑340 Project Two)**

This repository contains a Dash web application that visualizes the Austin Animal Center Outcomes dataset for Grazioso Salvare. The dashboard supports interactive filtering by rescue type, presents an interactive data table, and includes two synchronized visualizations: a geolocation map and a summary chart. These functions are demonstrated by the video linked below.

[Functionality Screencast](https://drive.google.com/file/d/1SZPgLdQH-AgaCRm-ZJOlvrFGd6bTl3IX/view?usp=sharing)

## **Required Functionality and Proof**

The dashboard allows users to filter the Austin Animal Center Outcomes dataset by rescue type, including Water Rescue, Mountain/Wilderness Rescue, and Disaster/Individual Tracking, with a Reset option that restores the full dataset. The interactive data table dynamically updates in response to the filters, supporting pagination, sorting, and filtering. Two charts also respond to the filtered dataset: a geolocation map that displays the selected animal’s coordinates and a bar chart showing the most common breeds. Branding requirements are satisfied with the Grazioso Salvare logo and the AJ identifier in the header.

## **Tools and Rationale**

The dashboard was developed using Python and Dash (JupyterDash) because Dash allows building interactive web dashboards directly in Python with clear callback structures that link user input to dynamic updates. dash‑leaflet was used for rendering the map with markers and popups, while Plotly Express provided easy‑to‑build charts that integrate seamlessly with Dash. pandas serves as the data manipulation layer, converting MongoDB results into tables suitable for visualization. MongoDB with PyMongo was chosen as the model component because of its flexible schema, powerful query operators, and ability to return data structures that map directly into pandas DataFrames. This makes it especially well suited for a dataset like Austin Animal Center’s, which has semi‑structured records and diverse attributes.

## **Resources and References**

The following tools and documentation were essential for this project: Dash documentation at https://dash.plotly.com/, Plotly Express guides at https://plotly.com/python/plotly-express/, the dash‑leaflet component library, pandas documentation at https://pandas.pydata.org/, and MongoDB resources at https://www.mongodb.com/ together with the PyMongo driver documentation at https://pymongo.readthedocs.io/.

## **Steps to Reproduce**

To reproduce the dashboard, set up a Python environment (version 3.9 or higher) and ensure that the Austin Animal Center dataset is loaded into MongoDB under the AAC.animals collection. Place the required files in the same directory: animal\_shelter.py containing the AnimalShelter class, the dashboard script (FinalDashboard.py), and the Grazioso Salvare logo image file (grazioso\_logo.png). Install dependencies with:

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

Then run the script:

python ProjectTwoDashboard.ipynb

The dashboard will launch showing the rescue type filters, the interactive data table, and the two charts side by side. Each rescue type filter corresponds to a MongoDB query, such as Labrador Retriever Mix, Chesapeake Bay Retriever, and Newfoundland for Water Rescue, German Shepherd, Alaskan Malamute, Siberian Husky, and Rottweiler for Mountain/Wilderness Rescue, and Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, and Rottweiler for Disaster/Tracking. Reset clears the filter and reloads the full dataset.

## **Challenges and Resolutions**

Several challenges were encountered during development. One issue was MongoDB returning an ObjectId in the \_id field, which caused errors when binding data to the Dash DataTable; this was resolved by dropping the \_id column. Early code also relied on hard‑coded column indices for latitude and longitude, which proved fragile, so the callbacks were updated to reference columns by name. Another challenge was handling missing or blank values, which was resolved with default values and checks in the callbacks. Ensuring consistency between the table, chart, and map required careful use of the same filter state across all components. Finally, handling branding requirements was addressed by encoding the provided logo as base64 for reliable display.

## **Maintenance Notes**

If the dataset schema changes, the callback functions may require updates to the column references (for example, breed, outcome\_type, name, location\_lat, location\_long). The second chart can be swapped to show outcome types or age distribution by modifying the graph callback. Connection settings and credentials are managed in animal\_shelter.py and should be updated for different environments.

## **License**

SNHU