**Author:** Francisco Sousa

**CS 340 – Project Two: README**

**Client:** Grazioso Salvare

**Instructor:** Helen Gebre-Amlak

**Date:** October 19, 2025

## **Overview and Functionality**

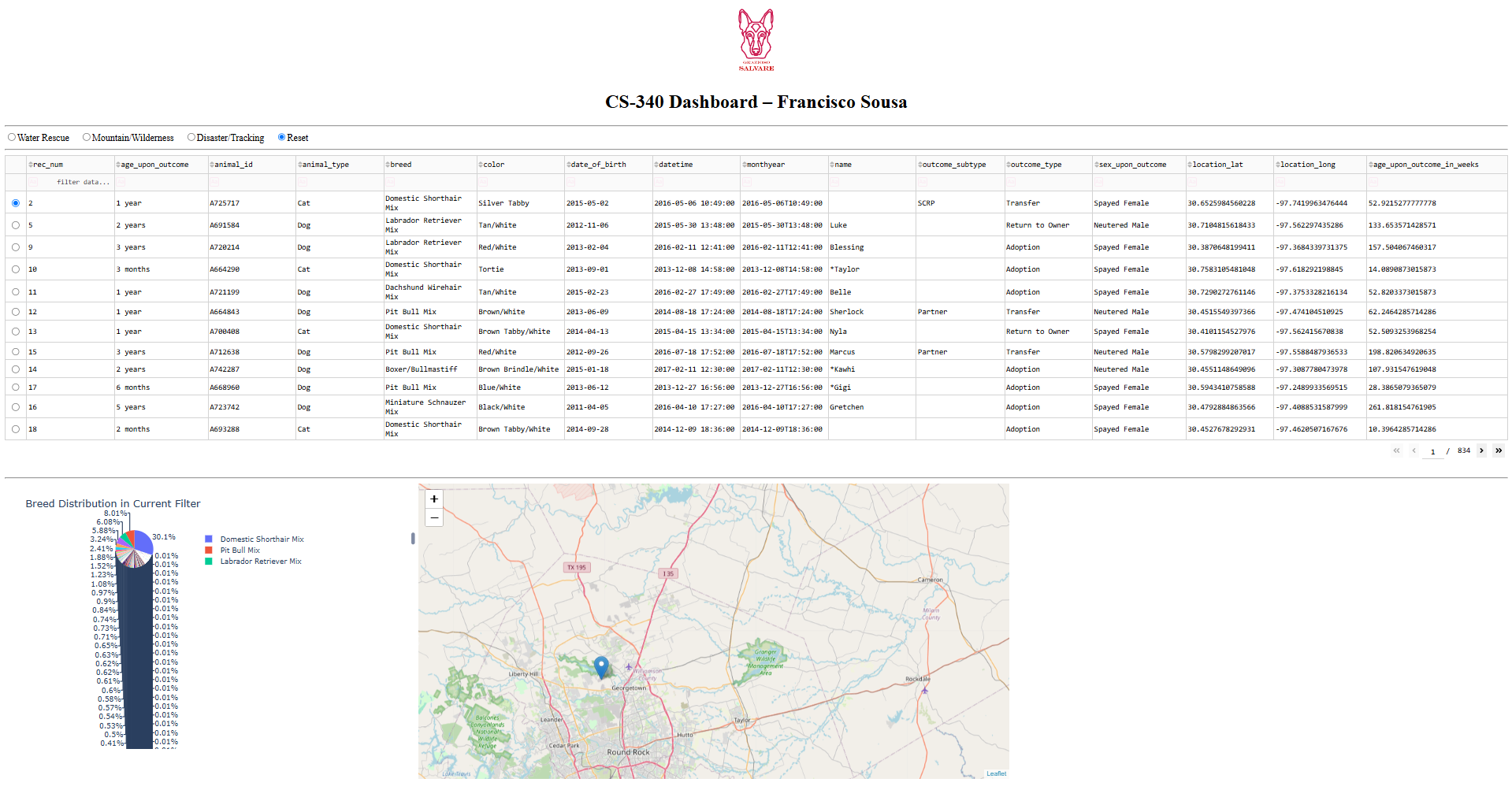
This project delivers a MongoDB-backed, interactive dashboard for Grazioso Salvare, a rescue-dog training organization.  
 The dashboard enables the client to identify dogs that meet search-and-rescue criteria across the Austin Animal Center Outcomes dataset.

The completed dashboard provides:

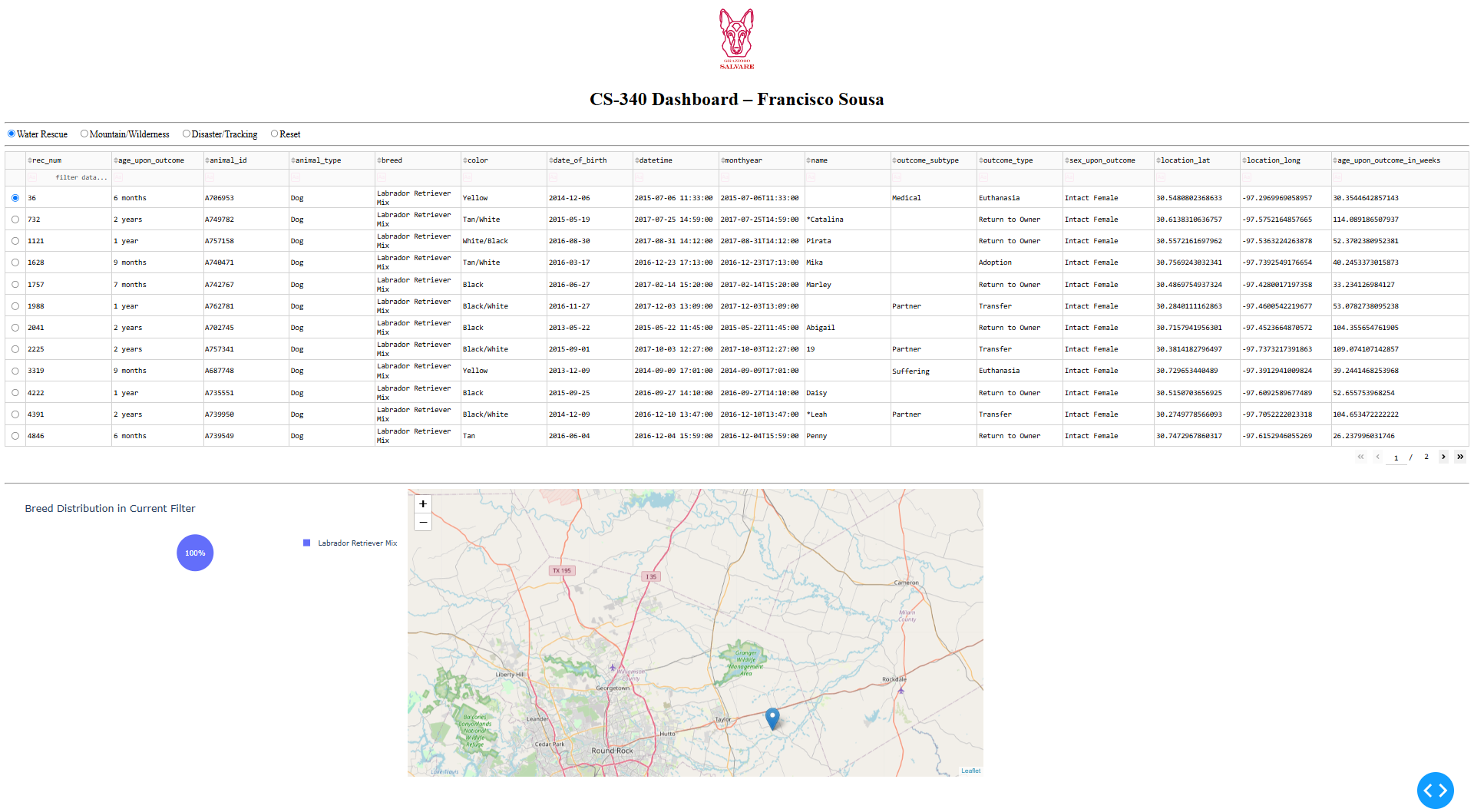
* **Interactive filters** (radio buttons) for  
   – Water Rescue  
   – Mountain / Wilderness Rescue  
   – Disaster / Individual Tracking  
   – Reset
* **Interactive Data Table** that refreshes automatically using MongoDB queries.
* **Two dynamic charts:**  
   – A Plotly pie chart of breeds represented in the filtered results.  
   – A Leaflet-based geolocation map that centers on Austin TX and plots the selected animal’s coordinates.
* **Branding:** The Grazioso Salvare logo and a unique identifier line: “Dashboard by Francisco Sousa.”

Each screenshot below demonstrates the required functionality.

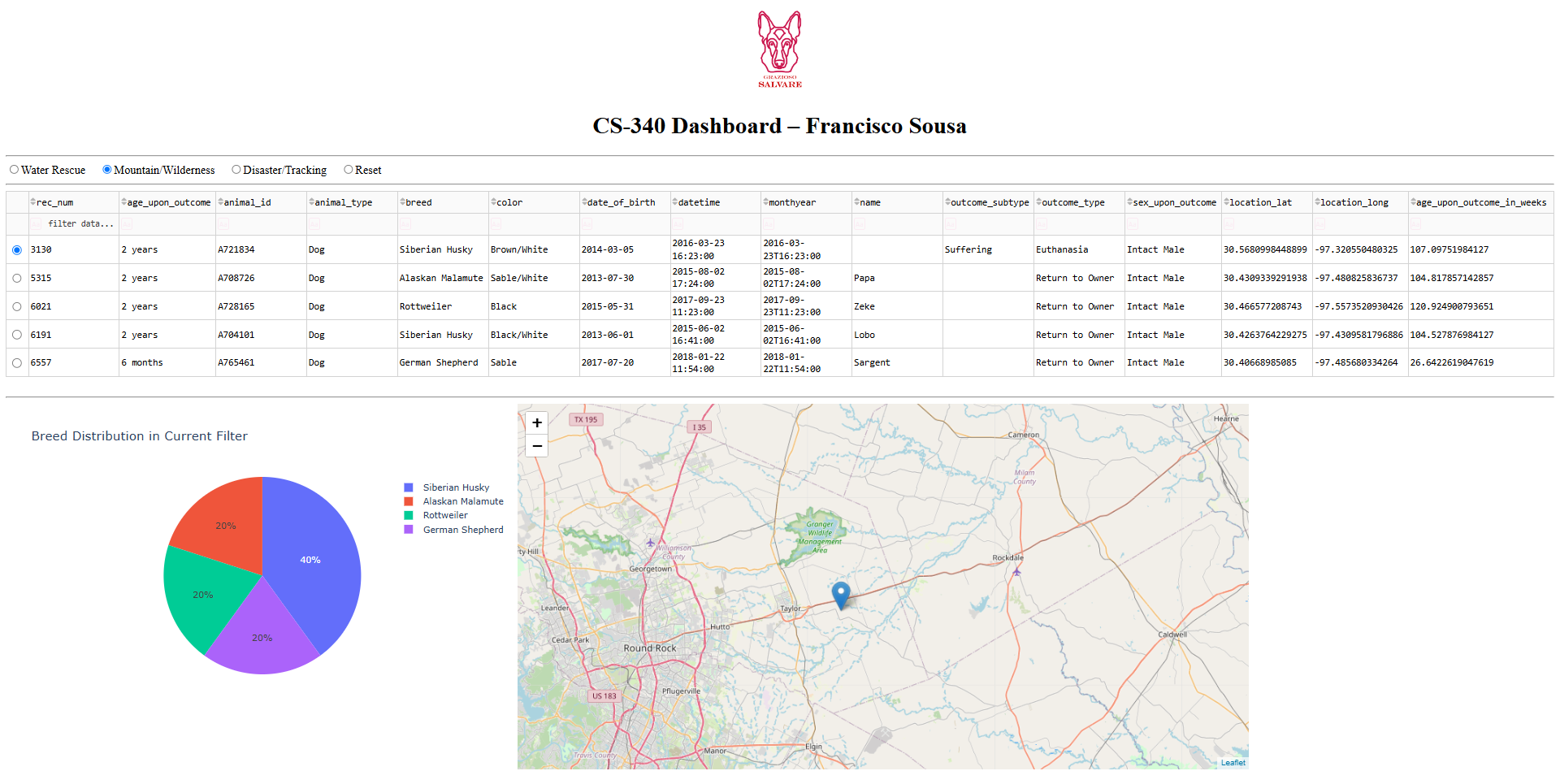
**Proof of Execution**



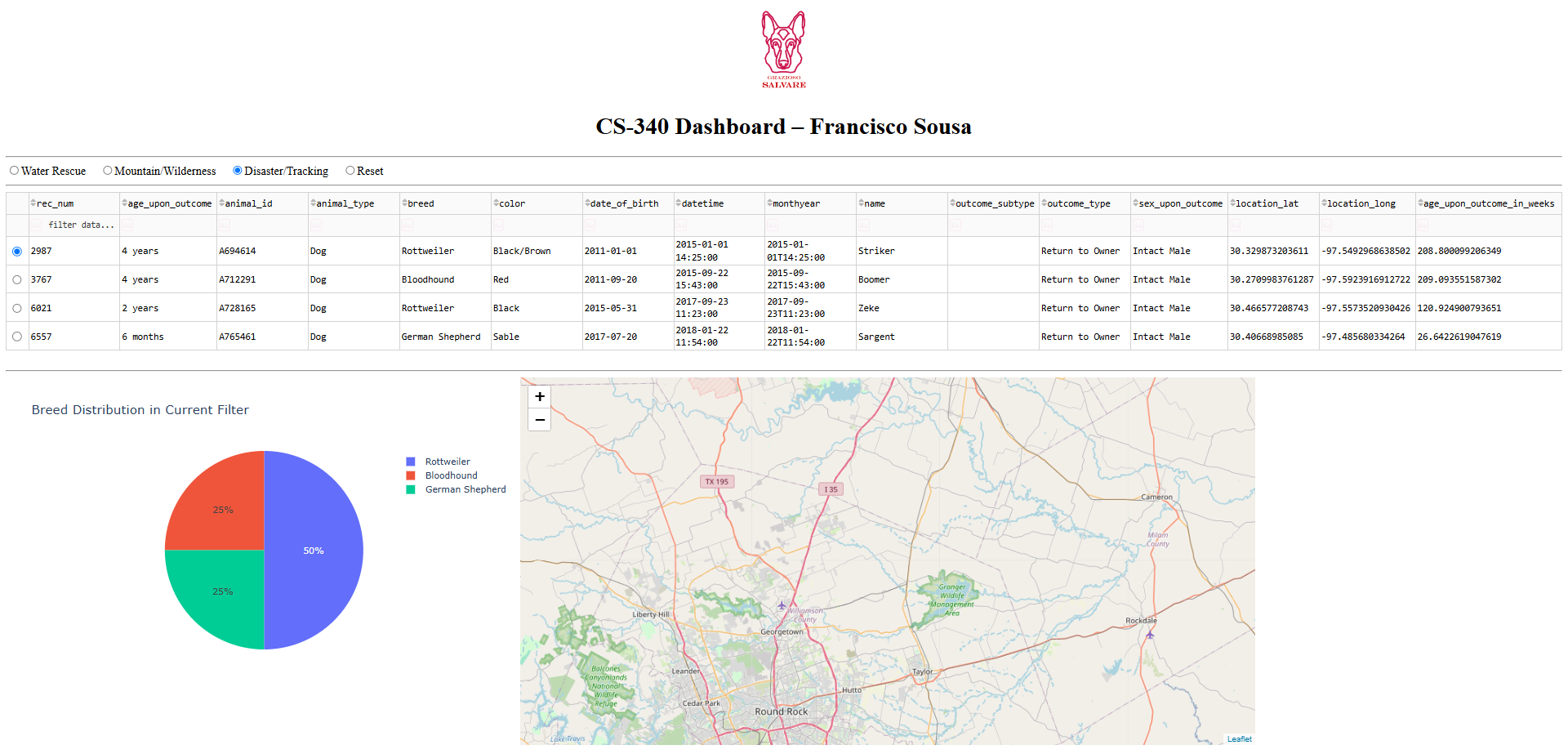
***Figure 1.*** *Initial State*



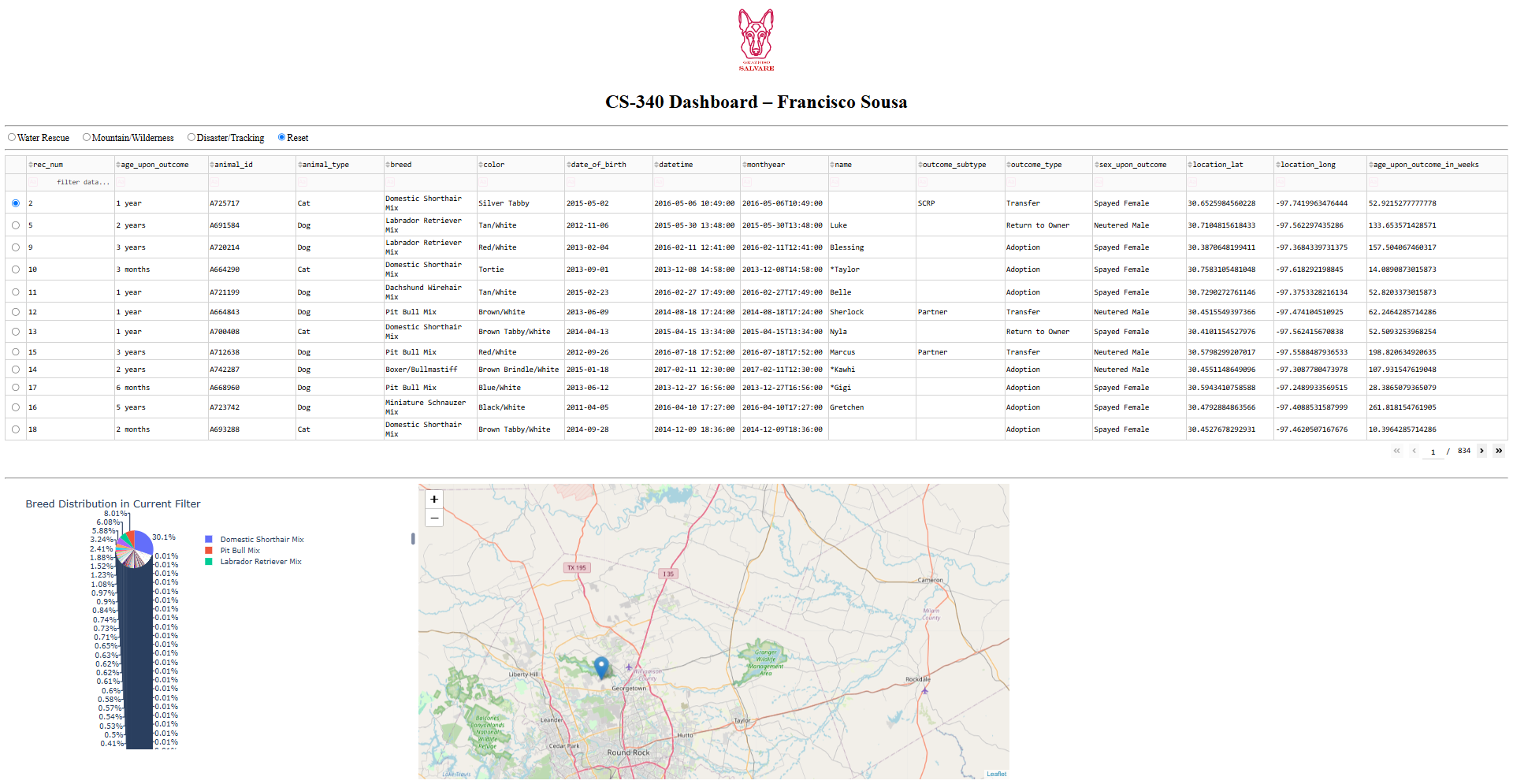
***Figure 2.*** *Water Rescue Filter*



***Figure 3.*** *Mountain / Wilderness Rescue*



***Figure 4.*** *Disaster / Tracking*



***Figure 5.*** *Reset*

**Tools and Technologies Used**

|  |  |  |
| --- | --- | --- |
| **Component** | **Purpose** | **Rationale** |
| MongoDB (Atlas / Codio local) | Database Model (MVC “Model”) | Schema-less structure stores varied animal records efficiently. Native JSON format maps directly to Python dict objects. |
| PyMongo and AnimalShelter CRUD Module | Secure CRUD interface | Encapsulates database credentials and operations (create, read, update, delete) for clean controller calls. |
| Dash / JupyterDash Framework | Web Application MVC View + Controller | Dash simplifies building interactive dashboards entirely in Python. Supports callbacks that link controls to data components. |
| Dash Core Components & DataTable | UI elements and interactive filters | Enables native sorting, paging, and filtering. |
| Dash Leaflet | Geolocation visualization | Provides real map tiles and marker overlays for selected dogs. |
| Plotly Express | Pie chart visualizations | Easily renders responsive charts that update with callbacks. |
| Pandas | Data wrangling layer | Converts MongoDB records to DataFrame objects for display. |
| Python 3 / Jupyter Lab on Codio | Development environment | Required virtual lab for SNHU CS 340 deliverables. |

**Why MongoDB as Model Component**

MongoDB is document-oriented and JSON-native, allowing direct mapping between Python objects and database records.  
 Its flexibility handles variable fields from different animal shelters without schema migrations.  
 Integration via PyMongo and the custom AnimalShelter class makes database queries simple to call from Dash callbacks.

**Why Dash for View and Controller**

Dash provides the MVC structure within Python:

* **View** → HTML and graphical layout (logo, buttons, tables, charts).
* **Controller** → @app.callback decorators that trigger database queries and chart updates.
* **Model** → MongoDB access through the CRUD module.

Dash simplifies data binding and client interaction without extra JavaScript.

## **Project Reproduction Instructions**

1. **Open Codio / Jupyter Lab.**
2. Place the following files in one folder:
   1. ProjectTwoDashboard.ipynb
   2. animal\_shelter.py (CRUD module)
   3. Grazioso Salvare Logo.png
3. Launch MongoDB and import the Austin Animal Center Outcomes.csv dataset.
4. Edit the credentials in the notebook (username = "aacuser", password = "SNHU1234").
5. Run each cell in order inside Jupyter Lab.
6. The dashboard will open on port 8050 (app.run\_server() in lab mode).
7. Select each radio filter to verify the table and charts update in real time.
8. Capture screenshots for submission as shown above.
9. Zip the project folder containing .ipynb, .py, and logo for upload.

**Challenges / Solutions**

|  |  |
| --- | --- |
| **Challenge** | **Resolution** |
| Handling MongoDB ObjectID column crashing the DataTable | Dropped the \_id field before loading data into the Pandas DataFrame to prevent Dash from reading non-serializable ObjectID objects. |
| Import errors for CRUD module | Renamed the CRUD file to animal\_shelter.py and matched the class name AnimalShelter to meet the dashboard import requirements (from animal\_shelter import AnimalShelter). |
| Callback error: “NoneType object is not iterable” | Occurred when no columns were selected in the DataTable. Added selected\_columns = selected\_columns or [] to safely handle None values. |
| Designing accurate filter logic for breed, sex, and age | Built queries using $and, $in, and range operators in MongoDB per the Rescue Type and Preferred Dog Breeds table from the Dashboard Specifications Document. |
| Maintaining dynamic links between table, pie chart, and map | Used derived\_virtual\_data as callback input to dynamically update both the Pie Chart and Leaflet Map whenever filters changed. |
| Port conflicts in Jupyter Dash | When Dash server ports were busy, used alternate ports or inline mode (app.run\_server(mode="inline")) for stable execution in Codio. |

## **References**

CS 340 Dashboard Specifications Document. Southern New Hampshire University, 2025.

CS 340 Module Six Dashboard Sample Walkthrough. SNHU course resource.

Plotly Documentation – Dash Core Components and DataTable.

Dash Leaflet Documentation (Plotly).

Giamas, A. (2022). *Mastering MongoDB 6.x – Advanced Querying.* Packt Publishing.

Make a README. SNHU course resource.

**End of README**