# **README – Grazioso Salvare Dog Candidate Dashboard**

**CS 340 – Project Two**  
**Author:** Naim Lindsay  
**Date:** August 16, 2025

## **About the Project**

This project delivers a **fully interactive dashboard** for **Grazioso Salvare**, a company that identifies and trains rescue dogs. The dashboard connects to a MongoDB database containing the **Austin Animal Center Outcomes** dataset and provides the following functionality:

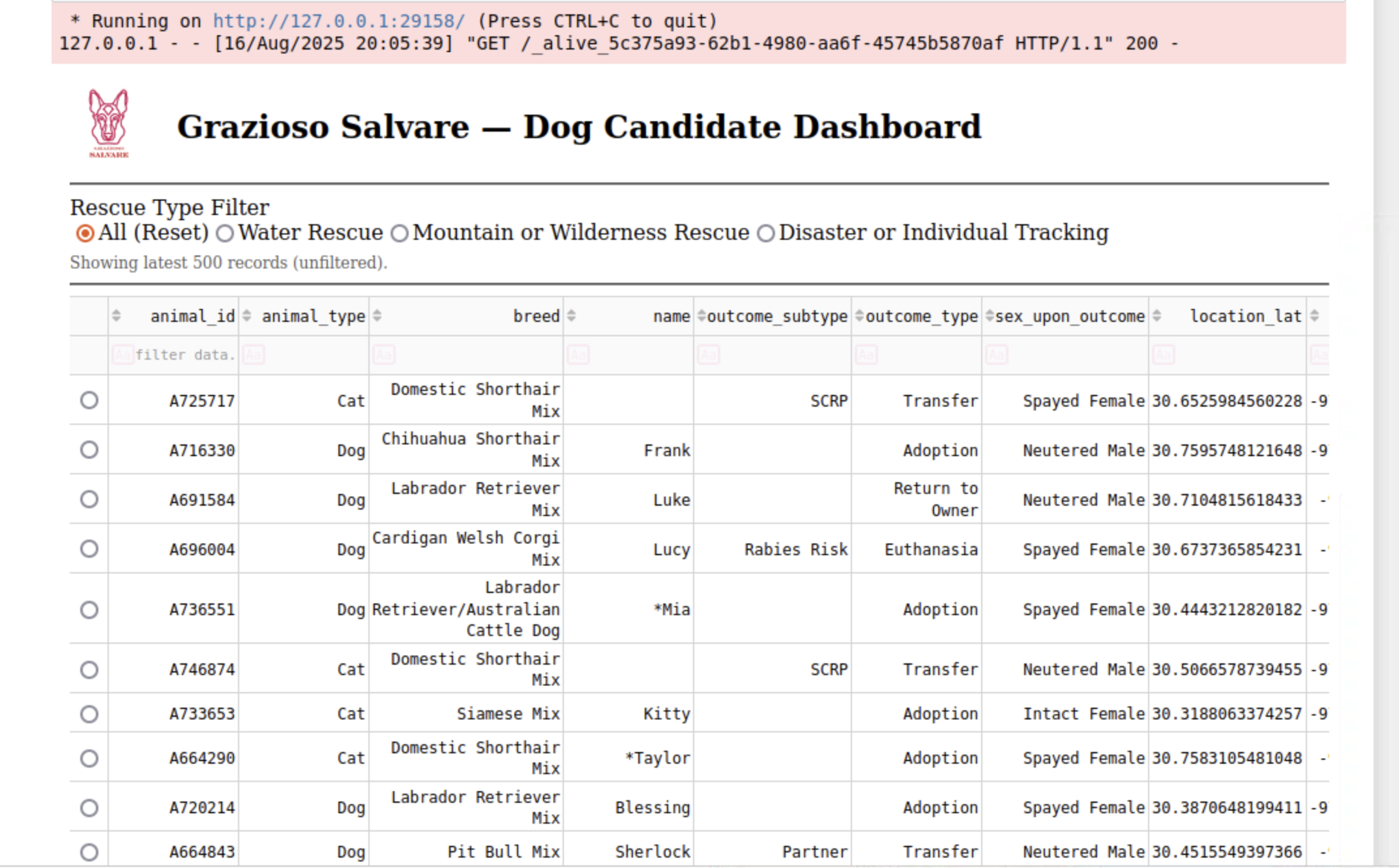
* **Rescue Type Filters**: Interactive radio buttons allow the user to filter dogs by:
  + Reset (All)
  + Water Rescue
  + Mountain or Wilderness Rescue
  + Disaster or Individual Tracking
* **Interactive Data Table**: Displays animal data pulled directly from MongoDB. Includes sorting, pagination, and filtering.
* **Dynamic Charts**:
  + **Pie Chart** – Displays the breed distribution for the currently selected filter.
  + **Geolocation Map** – Shows the locations of animals based on filter criteria.
* **Branding**: The dashboard includes the **Grazioso Salvare logo** and my **unique identifier** at the top.

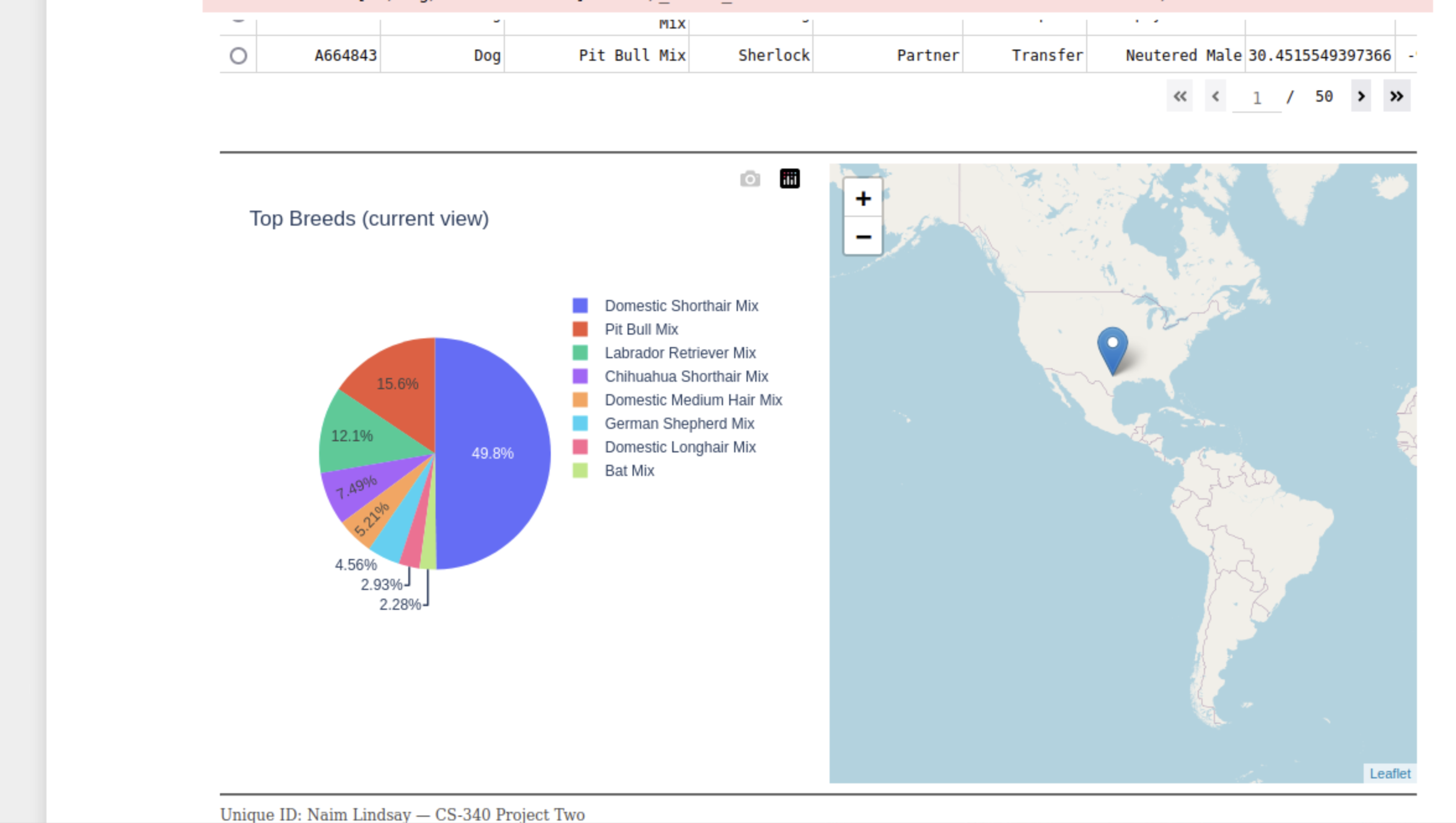
This system demonstrates how a real-world rescue organization can interactively explore animal data to identify the best candidates for training.

## **📸 Proof of Functionality**

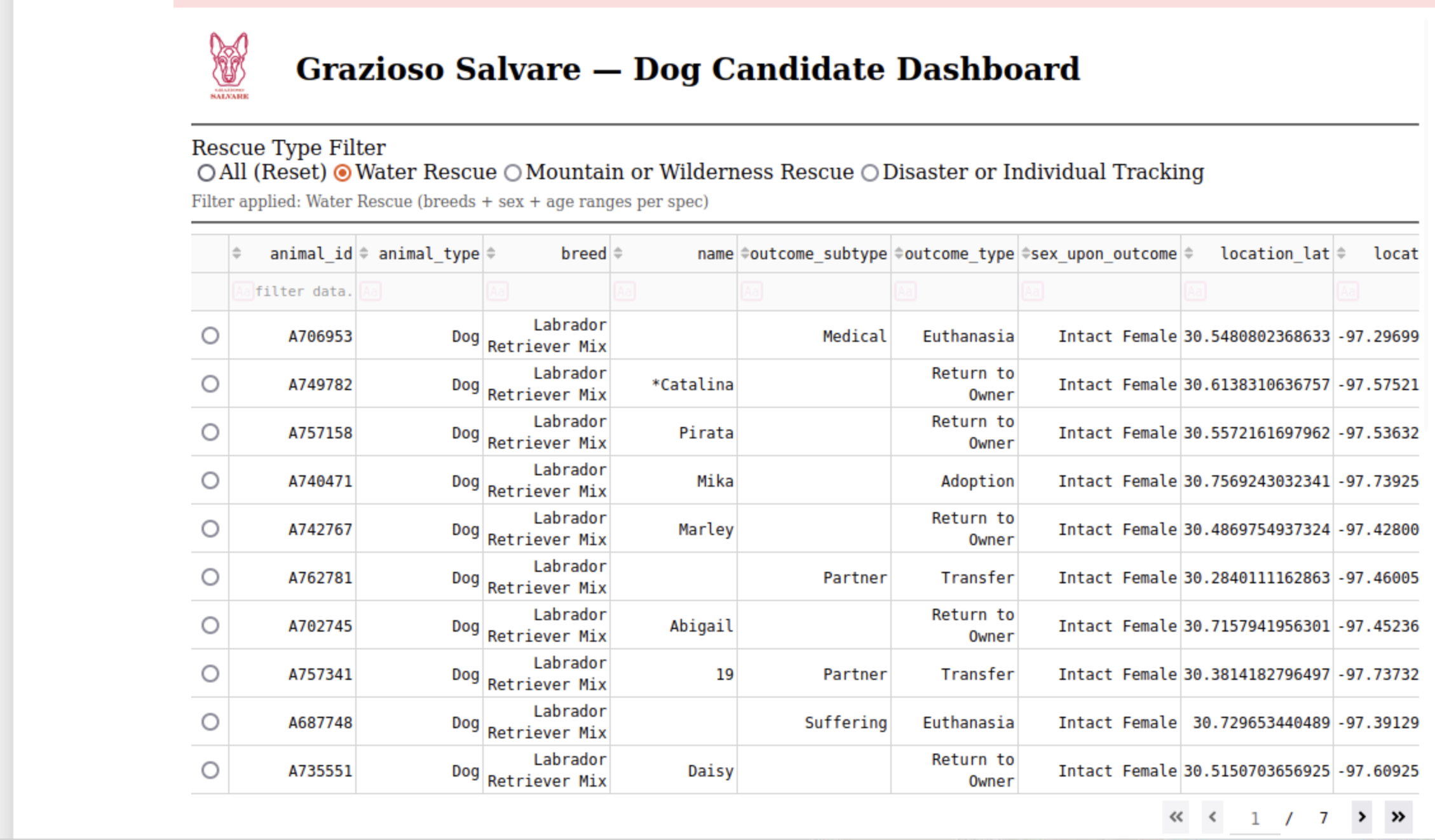
Below are the required screenshots that demonstrate the working dashboard:

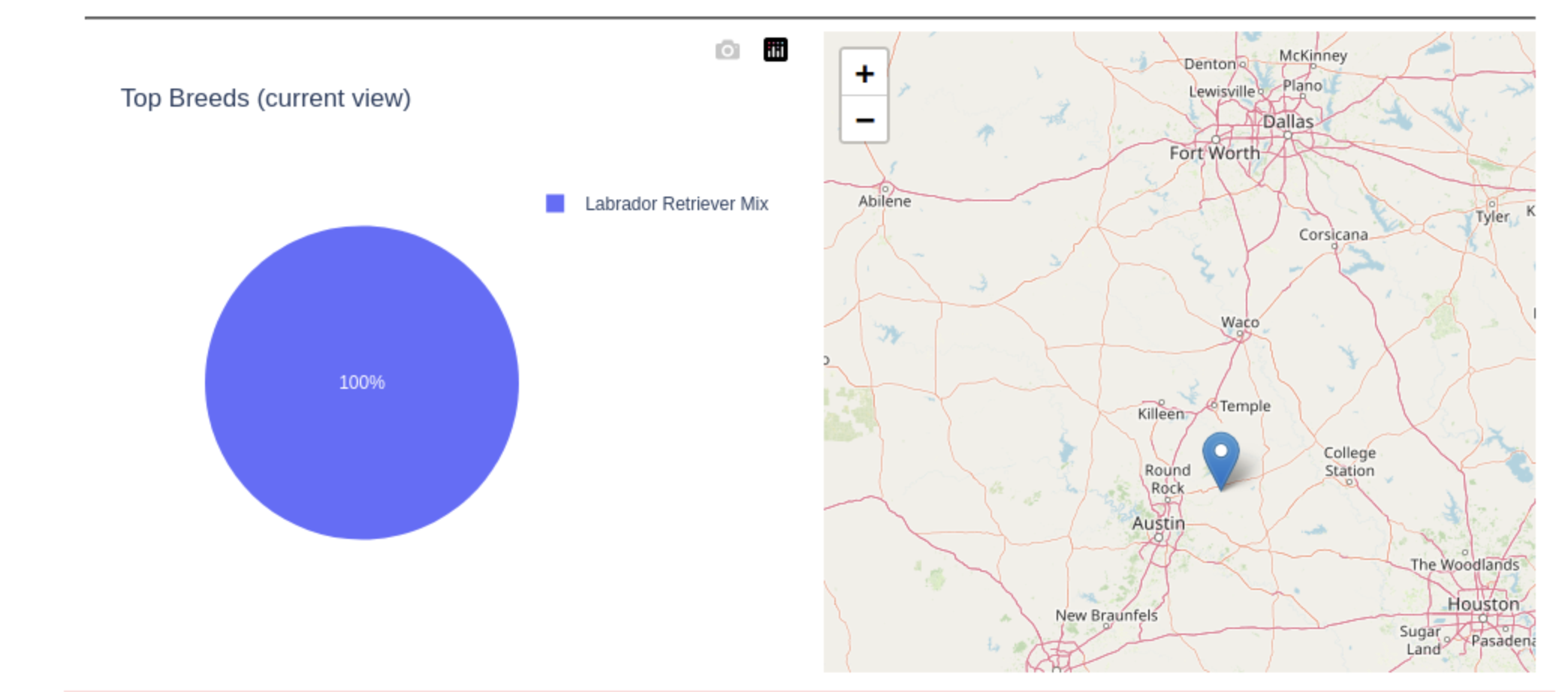
**Unfiltered Dashboard (All Selected)**  
*This screenshot shows the full Austin Animal Center dataset with no filters applied. My unique identifier is visible at the top along with the Grazioso Salvare logo.*



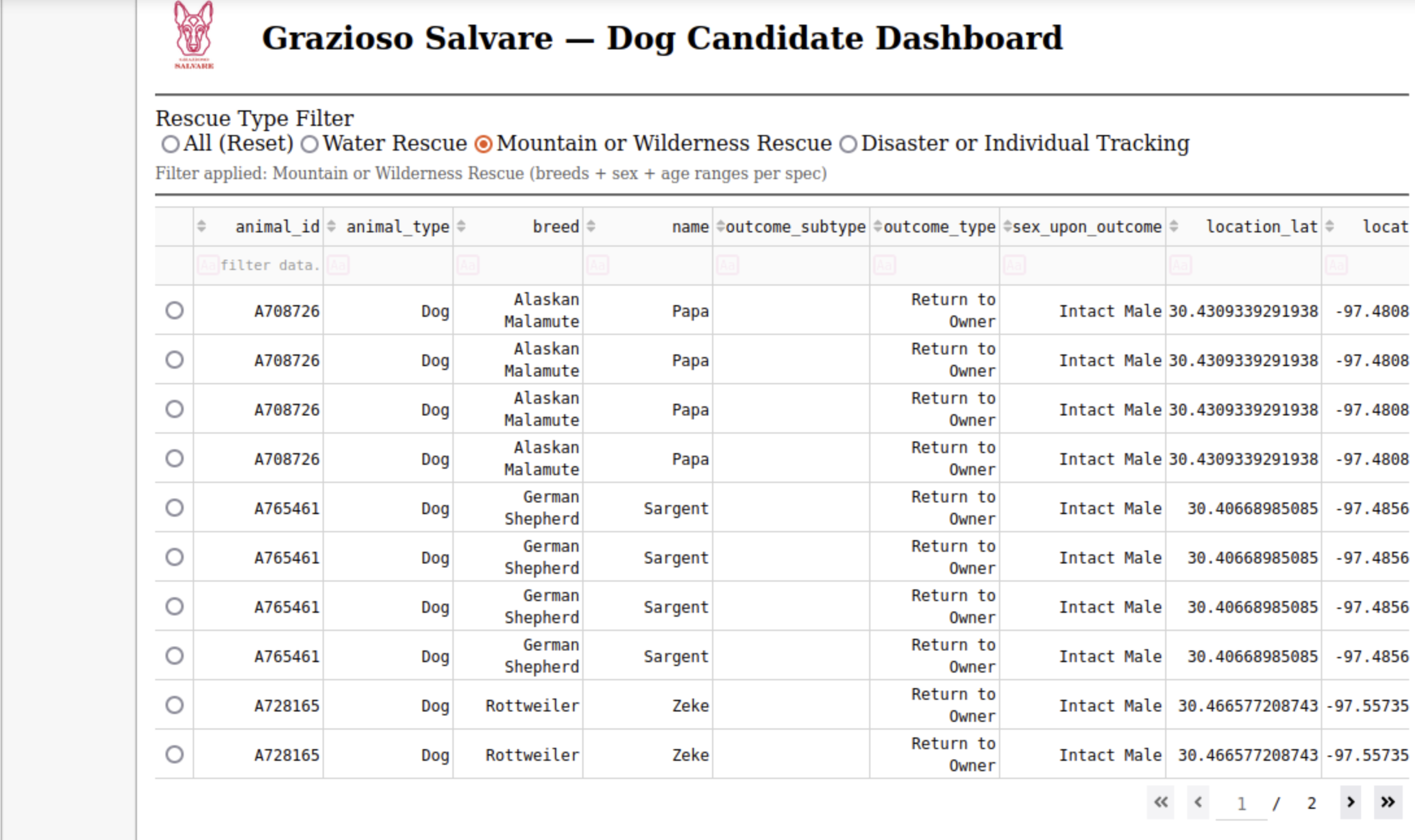


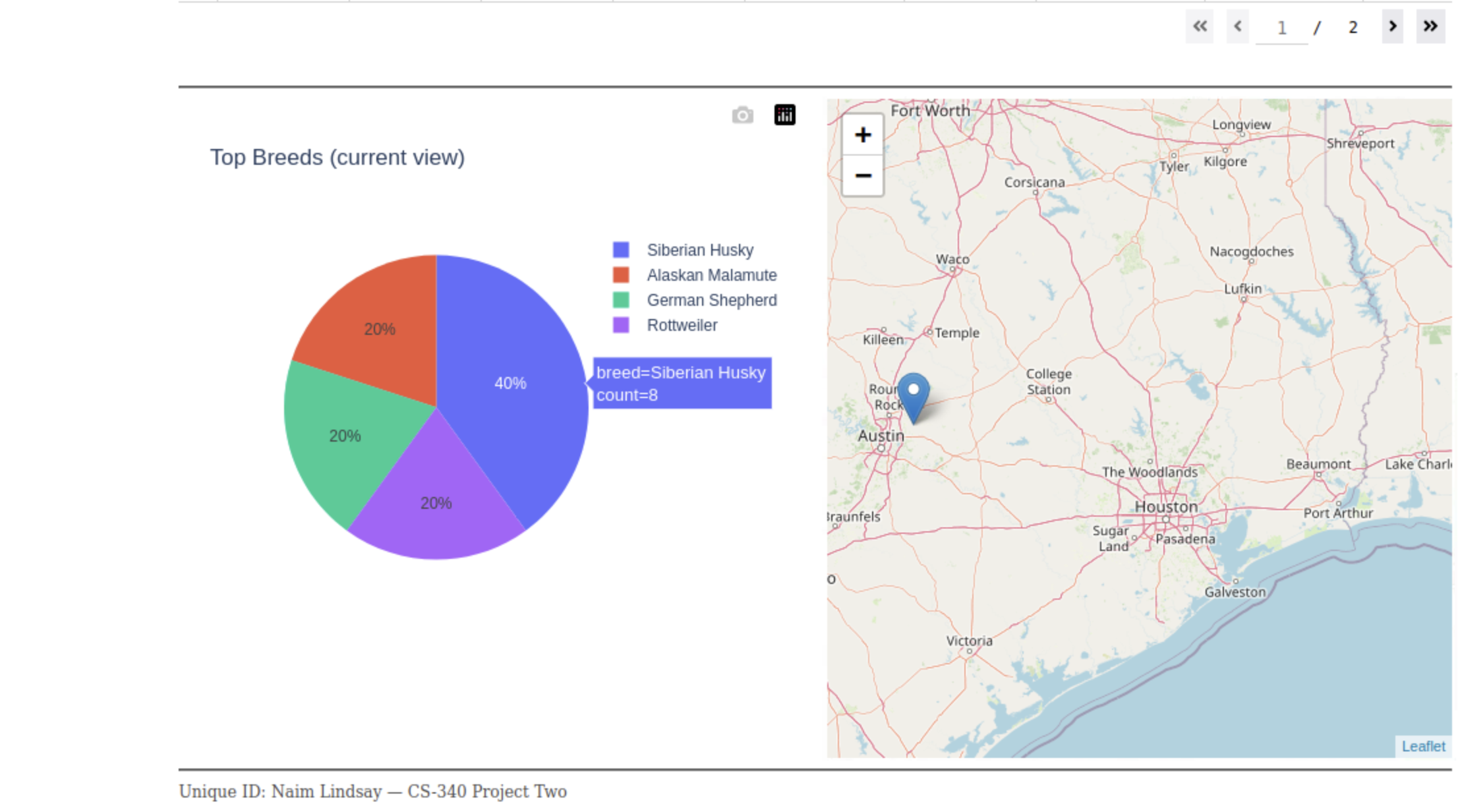
**Water Rescue Candidates**  
This screenshot shows the full Austin Animal/Reset . Center dataset with no filters applied. My unique identifier is visible at the top along with the Grazioso Salvare logo.





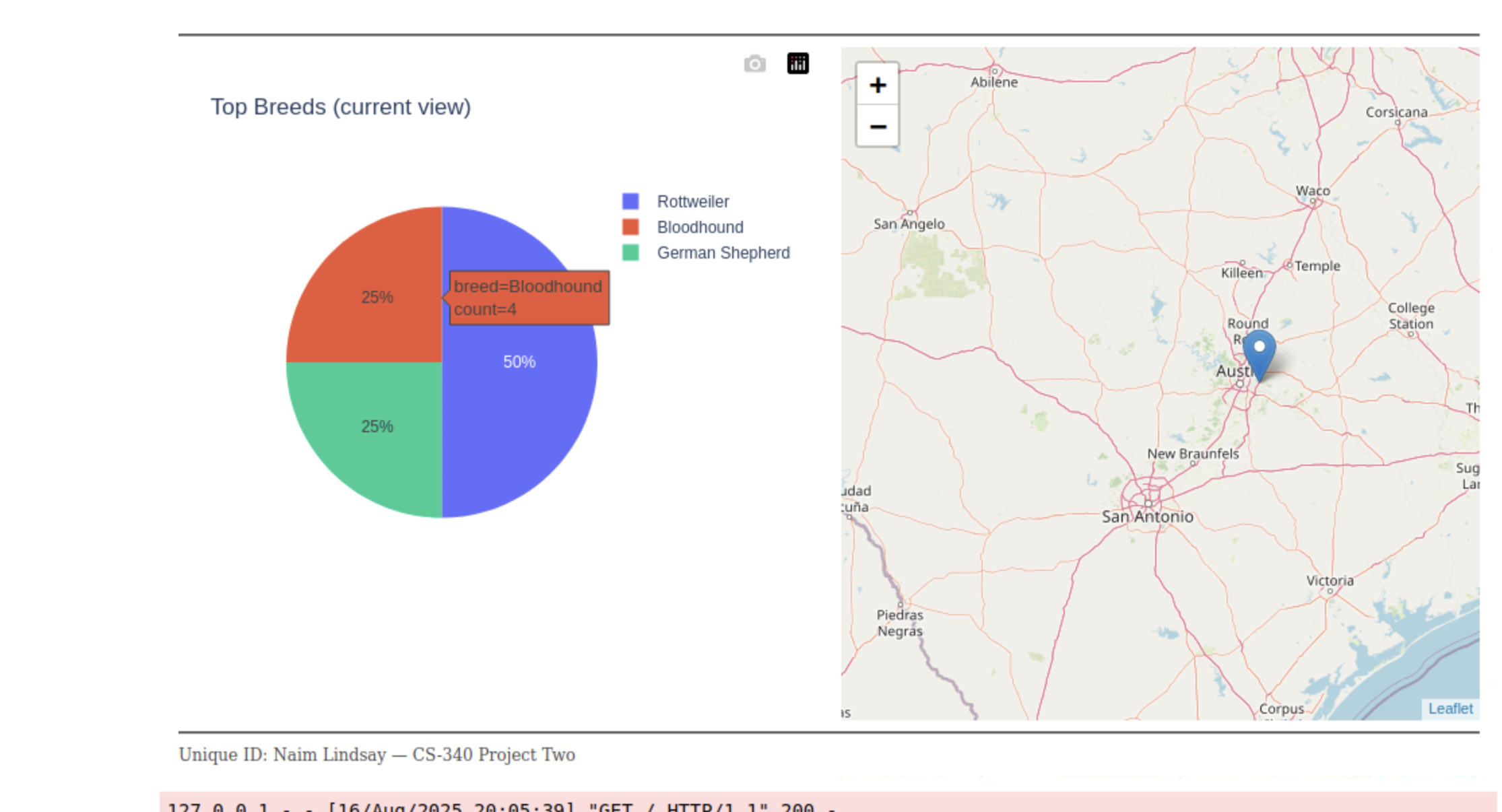
**Mountain or Wilderness Rescue Candidates**  
*This screenshot demonstrates the Mountain or Wilderness Rescue filter applied. The dashboard updates accordingly to show relevant rescue candidates.*





**Disaster or Individual Tracking Candidates**  
*This screenshot demonstrates the Disaster or Individual Tracking filter applied. The dashboard updates to show dogs that fit this category.*





## **⚙️ Tools Used**

* **MongoDB 6.0** – Model layer to store and query the Austin Animal Center dataset. Chosen for its flexibility in handling semi-structured JSON-like documents and seamless integration with Python through PyMongo.
* **PyMongo** – Python driver for MongoDB, used in the crud.py module to connect and query the database.
* **Dash & Plotly** – Controller and view framework. Dash powers the interactive web components (radio buttons, table, charts), while Plotly generates the visualizations.
* **Python 3 / Jupyter Notebook** – Development environment for building and testing the dashboard (ProjectTwoDashboard.ipynb).

## **Steps Taken**

1. Imported the crud.py module (from Project One) to handle MongoDB queries.
2. Created a **“retrieve all”** query to populate the base data table.
3. Built interactive radio button filters for **Rescue Type**.
4. Modified the data table so it updates dynamically when filters are applied.
5. Connected the filters to two charts:
   1. A **pie chart** of breed distribution.
   2. A **map visualization** of geolocation points.
6. Added branding: Grazioso Salvare logo and unique identifier.
7. Tested each filter state to confirm the table and charts updated correctly.

## **Challenges & Solutions**

* **Authentication Issues:** At first, connecting to MongoDB with a new user (aacuser) failed. I fixed this by creating the account properly and updating the connection string in crud.py.
* **Filter Logic:** Early queries returned no results because the breed/rescue filters didn’t match the dataset structure. I reviewed the **Rescue Type and Preferred Breeds table** from the specifications and corrected my queries.
* **Dashboard Layout:** Formatting the table and charts to display neatly was tricky. Dash’s built-in components (like pagination and responsive layouts) solved this.

## **🔗 Resources**

* [MongoDB Documentation](https://www.mongodb.com/docs/)
* Dash by Plotly Documentation
* PyMongo Documentation

## **✅ Conclusion**

This project demonstrates a full MVC architecture:

* **Model (MongoDB)** – Stores animal outcomes dataset.
* **Controller (Dash callbacks + CRUD module)** – Processes queries and filter logic.
* **View (Dash components + Plotly charts)** – Presents interactive results to the user.

The dashboard meets the client’s requirements by making the Austin Animal Center dataset **searchable, filterable, and visualized**, helping Grazioso Salvare quickly identify dog candidates for rescue training.