CS-340 – Project Two

Shawn Henly

04/11/25

## **README – Grazioso Salvare Rescue Dashboard**

### **About the Project**

This project presents an interactive data dashboard designed for Grazioso Salvare, a rescue-animal training company. The dashboard identifies adoptable dogs from the Austin Animal Center dataset who may be suitable for search-and-rescue training programs.

The dashboard was built using the MVC architecture:

* **Model:** MongoDB stores the AAC outcomes data.
* **View & Controller:** Dash components render the user interface and control data interactions.

### **Motivation**

Grazioso Salvare needed a tool that would allow them to view, filter, and analyze animal shelter records to select optimal dog candidates for training. The project serves as a client-facing solution that is user-friendly, secure, and adaptable for future enhancements. This dashboard enables non-technical staff to explore records interactively with no technical training required.

### **Getting Started**

To run the dashboard locally or in the SNHU Apporto environment:

1. **Log into Apporto.**
2. **Open a terminal** and run the following command to import the dataset:

mongoimport --username="aacuser" --password="CrossCrusaders707" --authenticationDatabase=admin --host=nv-desktop-services.apporto.com --port=30434 --db=AAC --collection=animals --type=csv --headerline --drop ./aac\_shelter\_outcomes.csv

1. Upload or copy your animal\_shelter.py file (CRUD module) into the same directory.
2. Open and run ProjectTwoDashboard.ipynb in Jupyter Notebook.
3. The dashboard will launch inline and allow filtering and data interaction.

### **Installation**

The following tools were used:

* **Python 3.9** (Apporto preinstalled)
* **MongoDB 5.0** – Flexible, document-based NoSQL database that integrates seamlessly with Python.
* **PyMongo** – MongoDB client library for Python used in the CRUD module.
* **Dash & JupyterDash** – Web framework for building dashboards within Jupyter.
* **Plotly Express** – Used to render dynamic visualizations (pie charts).
* **Dash Leaflet** – Used to create the geolocation chart for animal location data.

### **Usage**

The dashboard provides:

* A data table showing adoptable animals.
* A pie chart of breed distribution based on filters.
* A geolocation map pinpointing animal shelter coordinates.
* Filtering options via radio buttons:
  + Water Rescue
  + Mountain or Wilderness Rescue
  + Disaster or Individual Tracking
  + Reset

Code Example

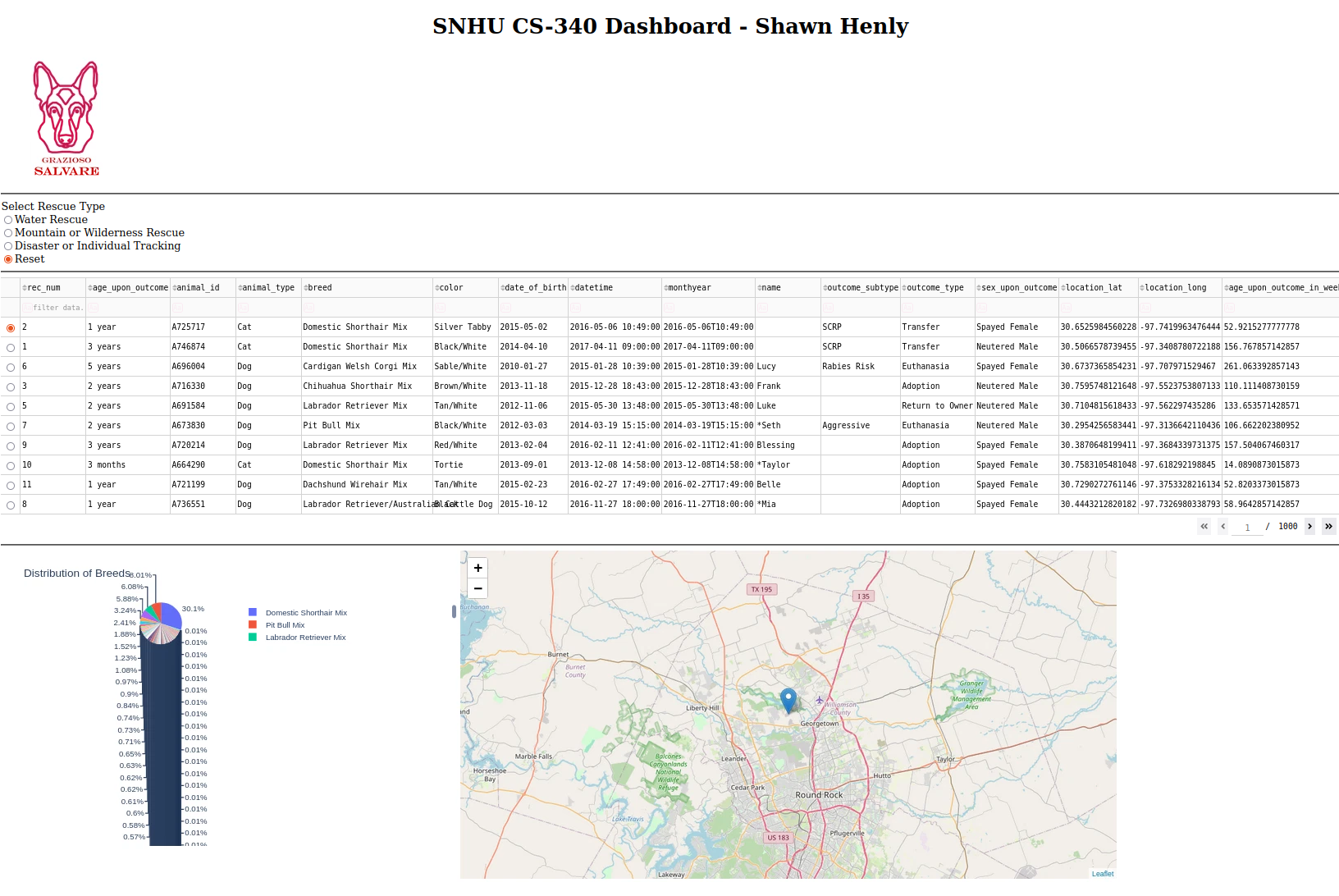
@app.callback(

Output('datatable-id', 'data'),   
 Input('filter-type', 'value')   
)   
def update\_dashboard(filter\_type):  
 if filter\_type == 'Water Rescue':  
 query = {...}   
 elif filter\_type == 'Mountain or Wilderness Rescue':  
 query = {...}   
 ...  
 result\_df = pd.DataFrame.from\_records(shelter.read(query))  
 ...  
 return result\_df.to\_dict('records')

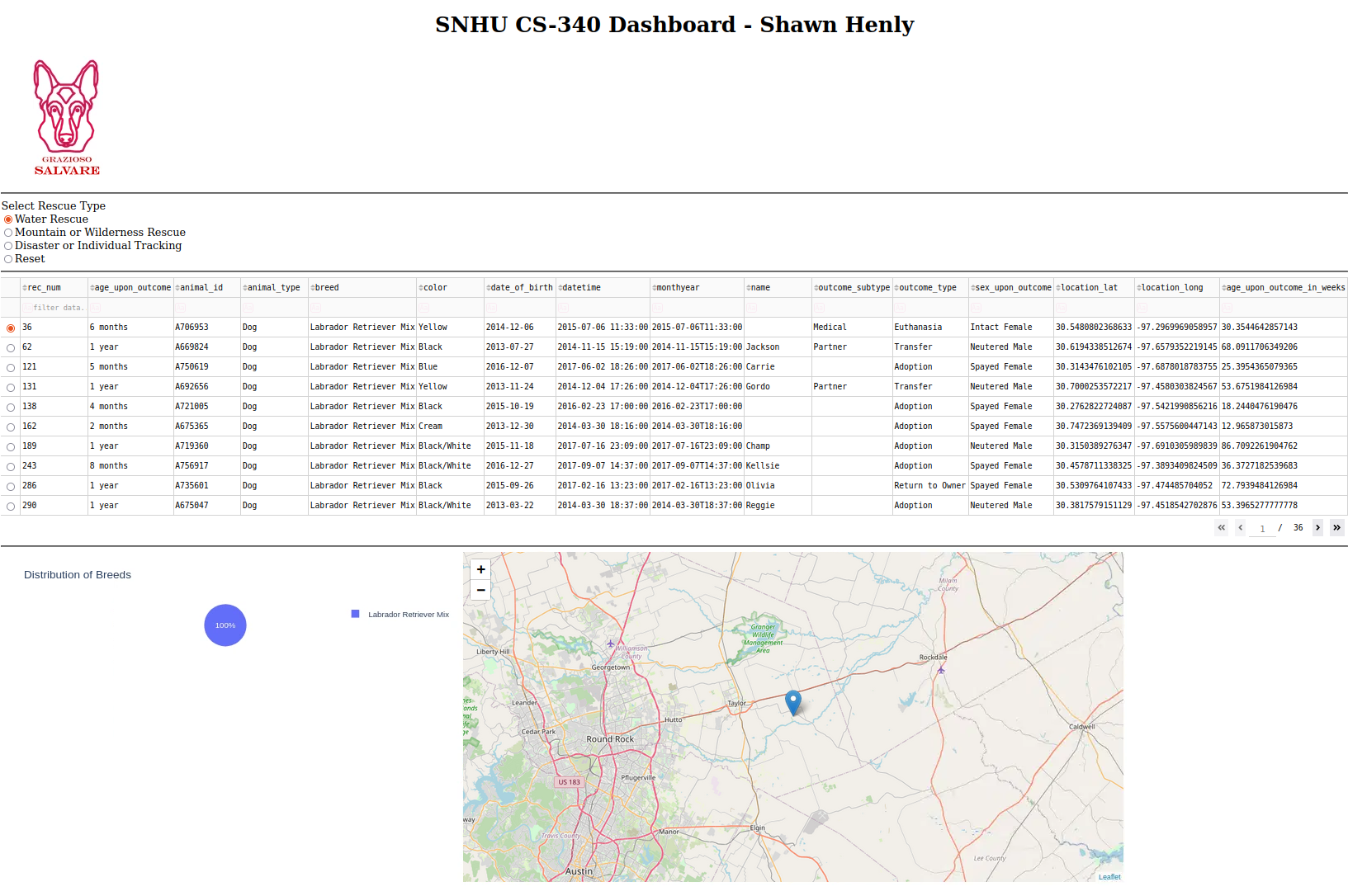
### **Tests**

All functionality was verified using manual execution of the dashboard. Screenshots below validate successful filtering and data updates:

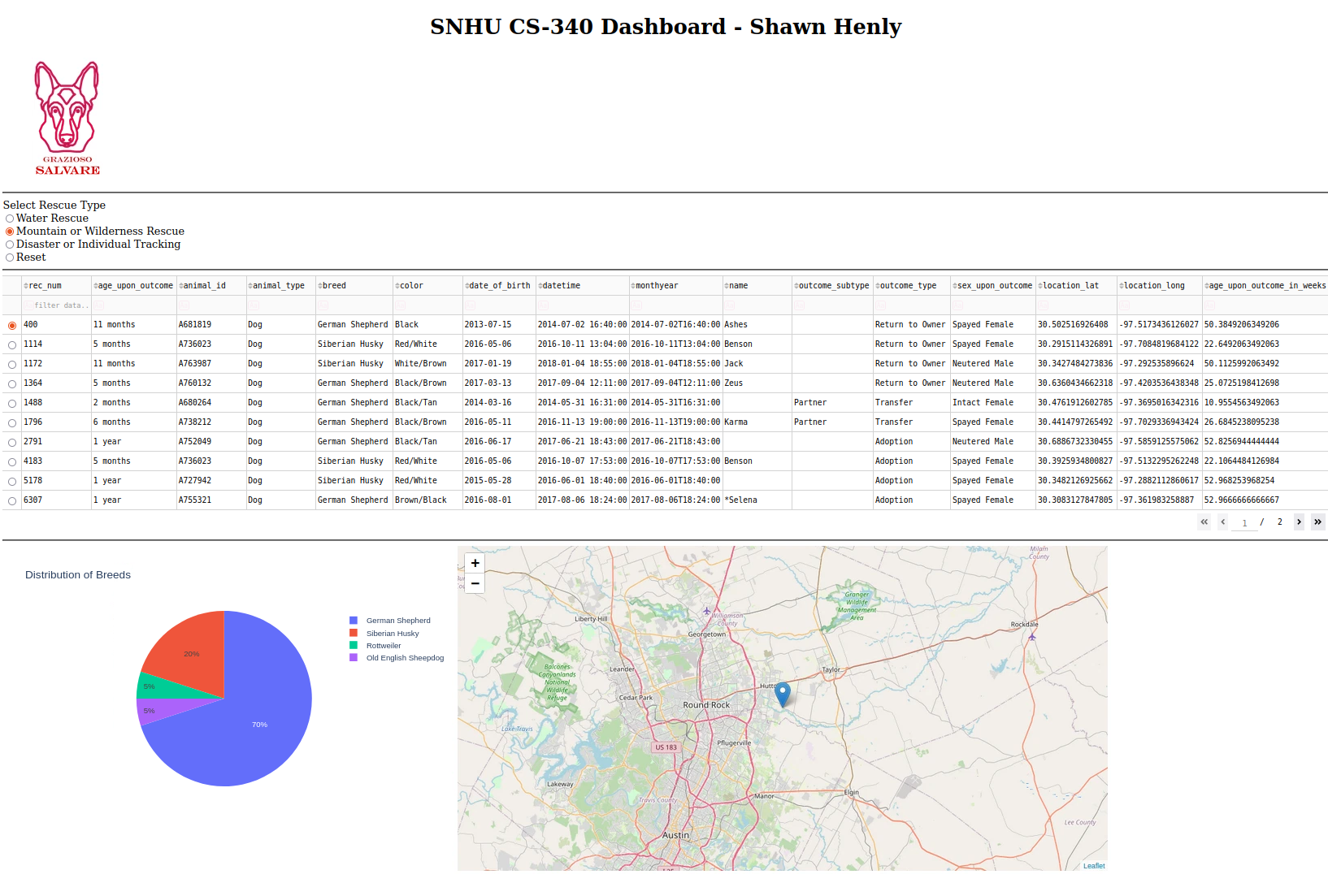
* **Screenshot 1:** Default dashboard view (unfiltered)



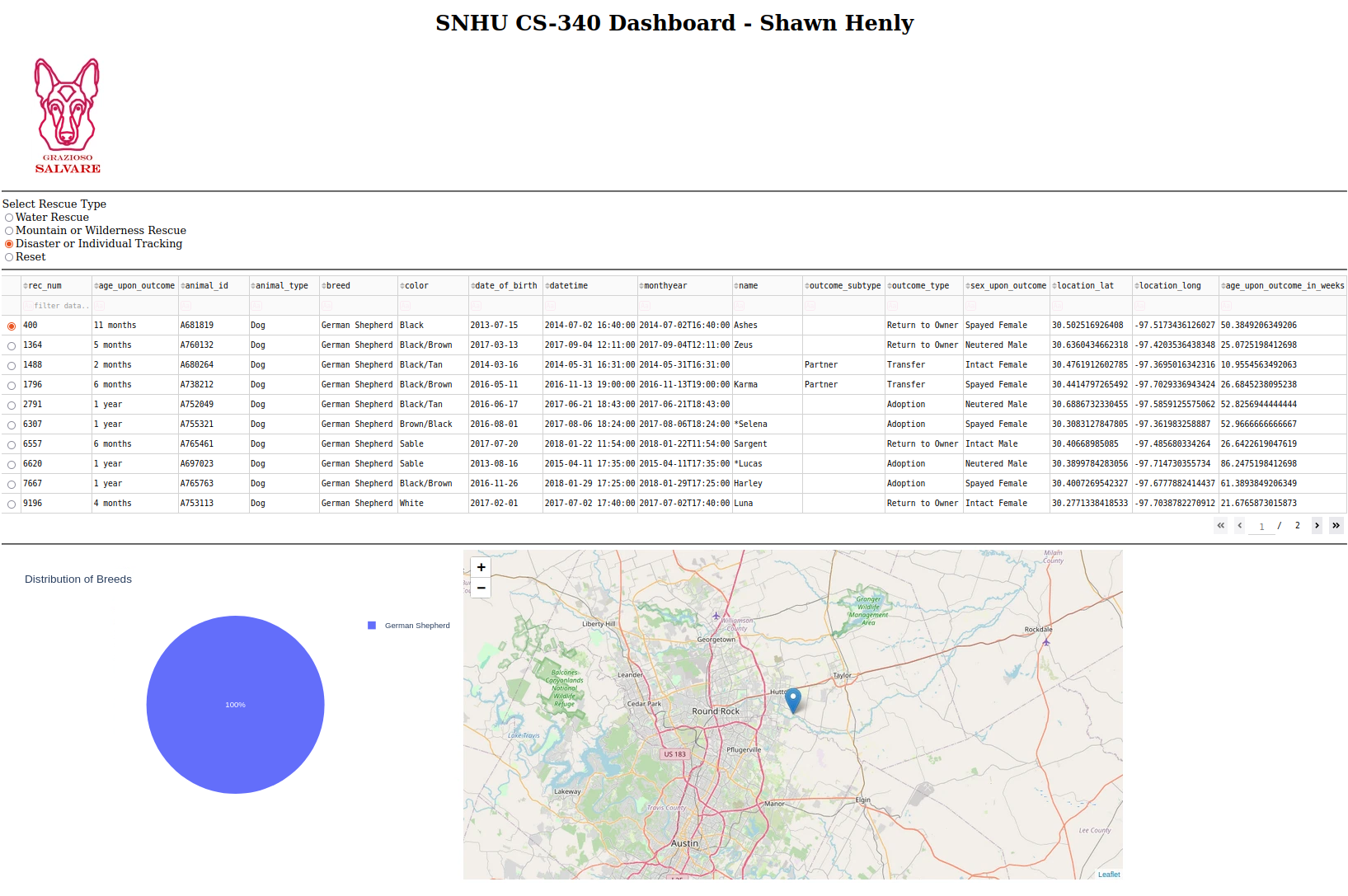
* **Screenshot 2:** Water Rescue filter applied



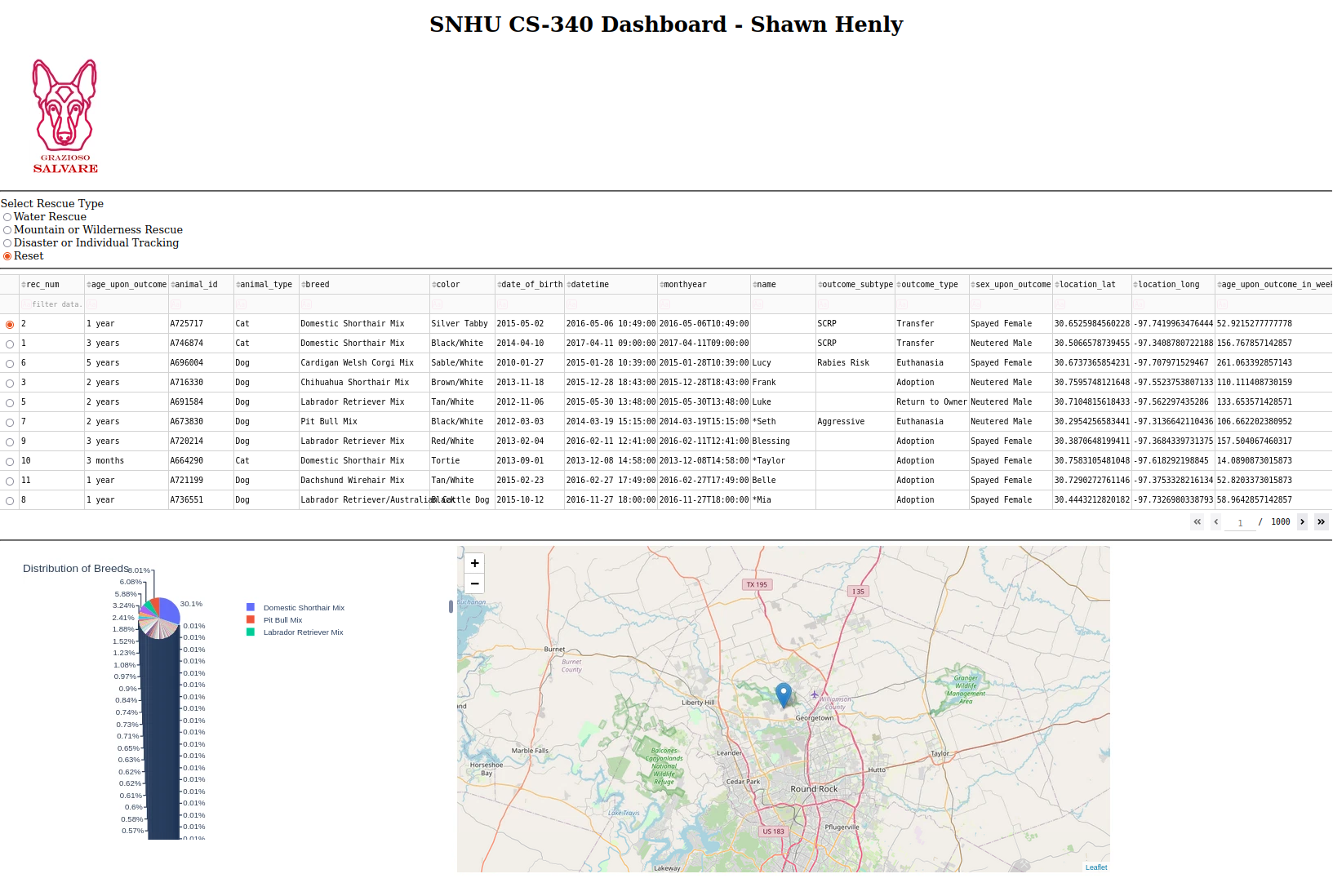
* **Screenshot 3:** Mountain or Wilderness Rescue filter applied



* **Screenshot 4:** Disaster or Individual Tracking filter applied



* **Screenshot 5:** Reset (all data restored)



### **Challenges and Solutions**

* **Map Marker Errors:** Some entries lacked geolocation. This was fixed using conditional logic to ensure safe access of location\_lat and location\_long.
* **CRUD Module Testing:** Initially, the read method was overridden. It was corrected to return a list of documents to work with pandas.DataFrame.
* **Pie Chart with No Data:** When filters returned zero records, the pie chart would crash. This was resolved by adding safe-checks to the callback.

### **Contact**

**Shawn Henly**  
[shawn.henly@snhu.edu](mailto:shawn.henly@snhu.edu)