Grazioso Salvare Rescue Dog Dashboard

Developer: Allen Falcon  
 Unique ID: 3232  
 Course: CS 340 – Client/Server Development

Project Overview

So this project develops a fully interactive MongoDB-backed web dashboard for Grazioso Salvare, a company that identifies and trains rescue dogs.

The dashboard lets users:

-Filter animals by rescue-type categories (Water, Mountain/Wilderness, Disaster/Individual Tracking, or Reset to view all).

-View an interactive data table of results pulled live from MongoDB.

-See a dynamic pie chart summarizing animal outcome types.

-Explore a geolocation map pinpointing the selected animal’s shelter coordinates.

So the application was built entirely in Python using the MVC design pattern:

Model: MongoDB database (aac.animals)

View: Dash dashboard widgets (DataTable, Map, Charts)

Controller: AnimalShelter CRUD Python module

Tools and Technologies Used

So, as we see and used through this class we use MongoDB we use it becuase Its flexible JSON-like documents easily store animal data with diverse attributes (breed, age, location). It integrates seamlessly with Python via pymongo.

Then Python + Dash / JupyterDash, Provides both View and Controller layers. Dash builds interactive web apps directly from Python — no JavaScript required. :)

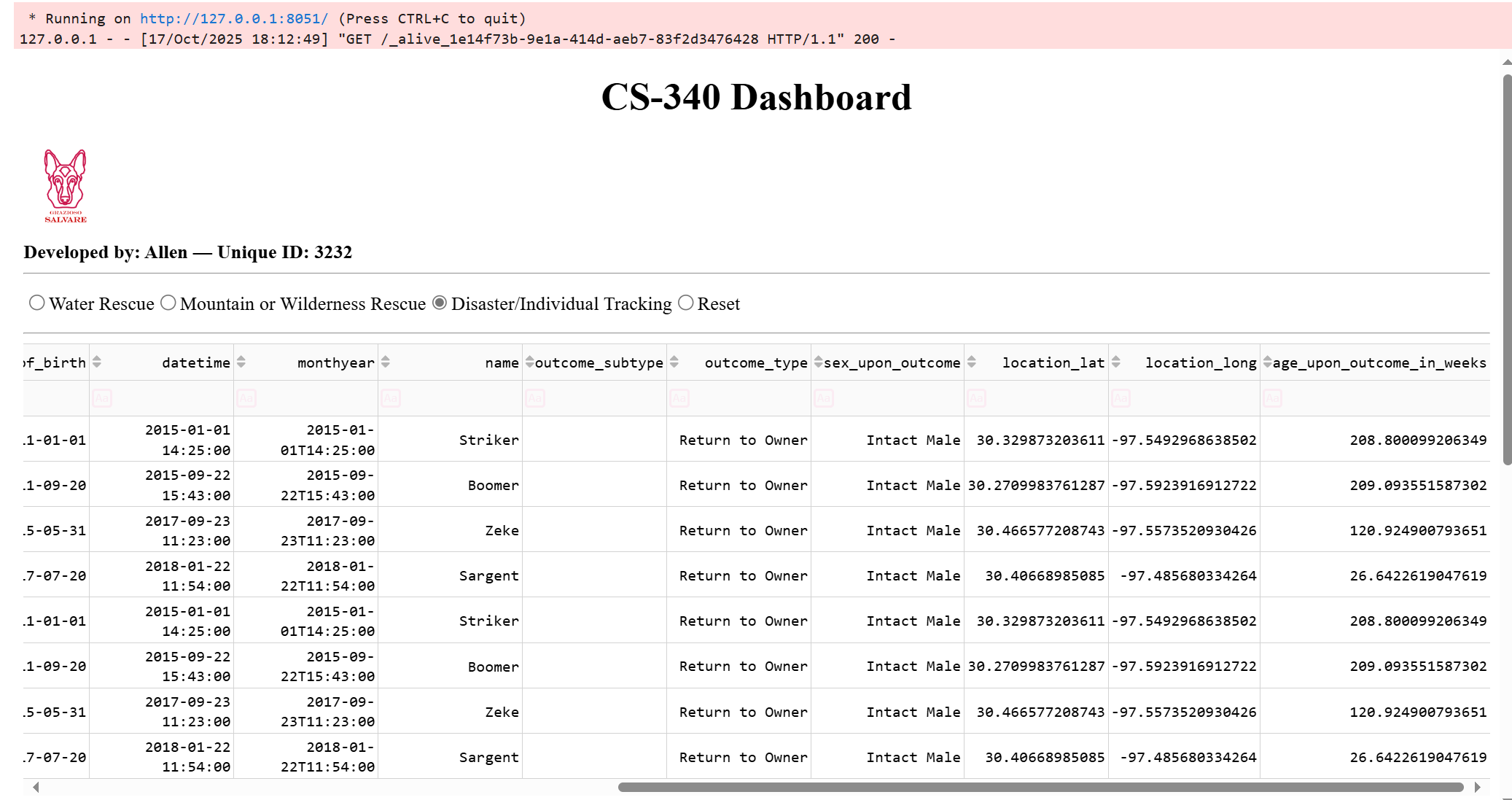
Plotly Express, Generates responsive charts (pie chart for outcome analysis).

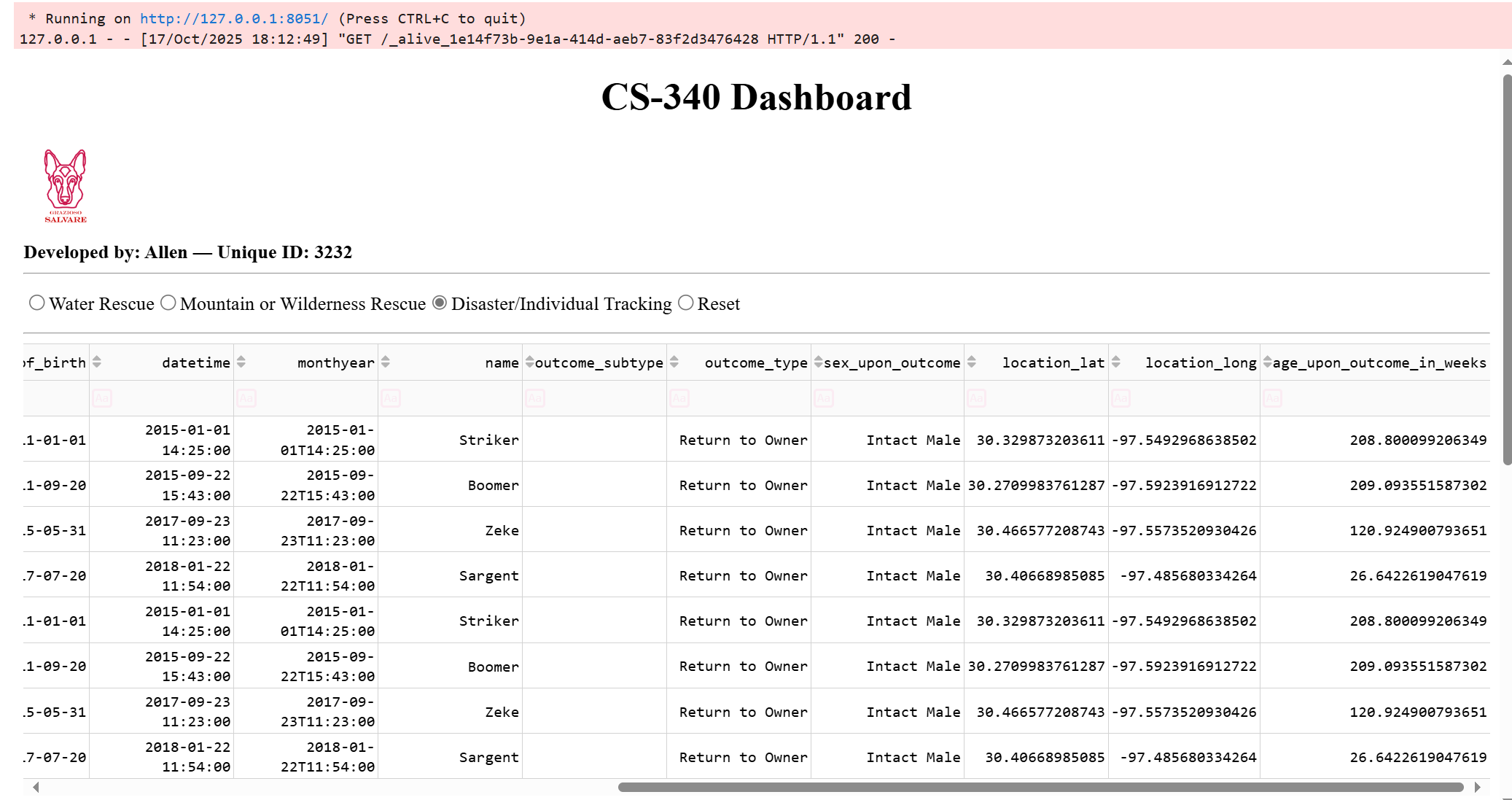
Dash, Displays a live interactive map with animal location markers.

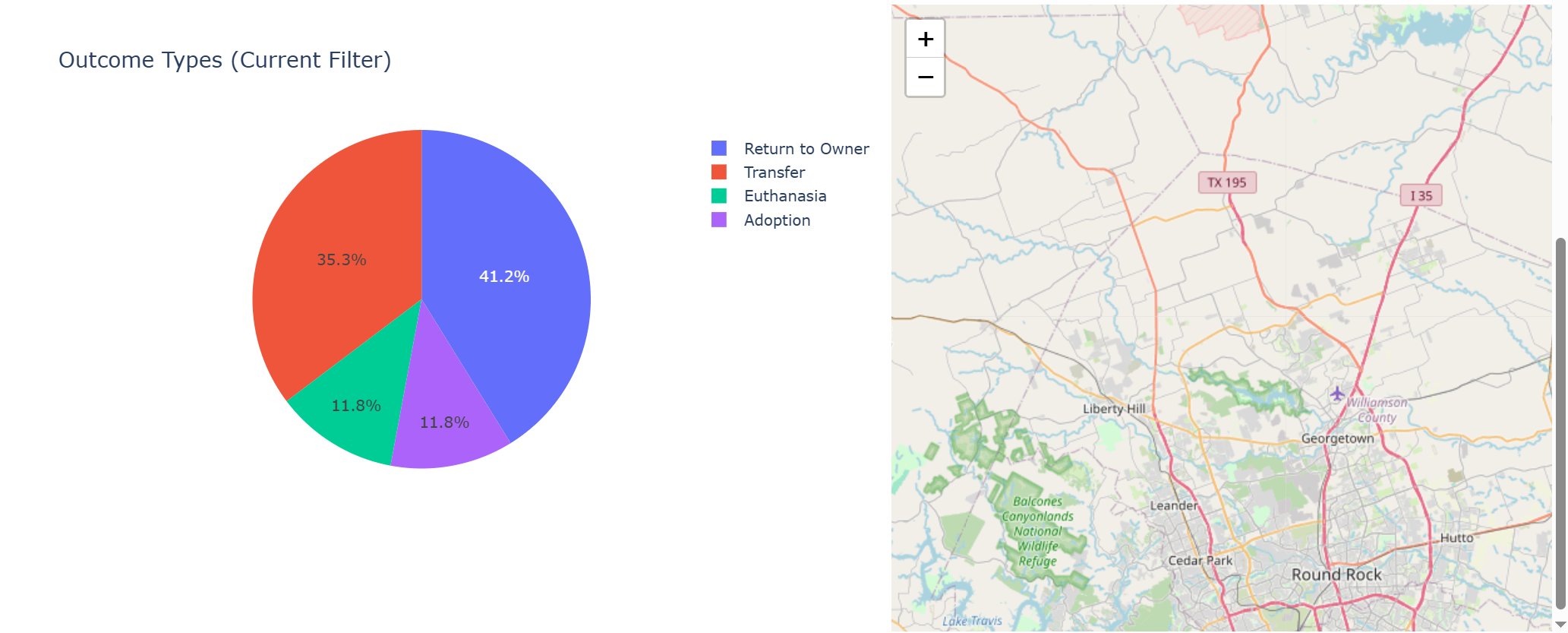
Pandas, Handles tabular data conversion between MongoDB and Dash.

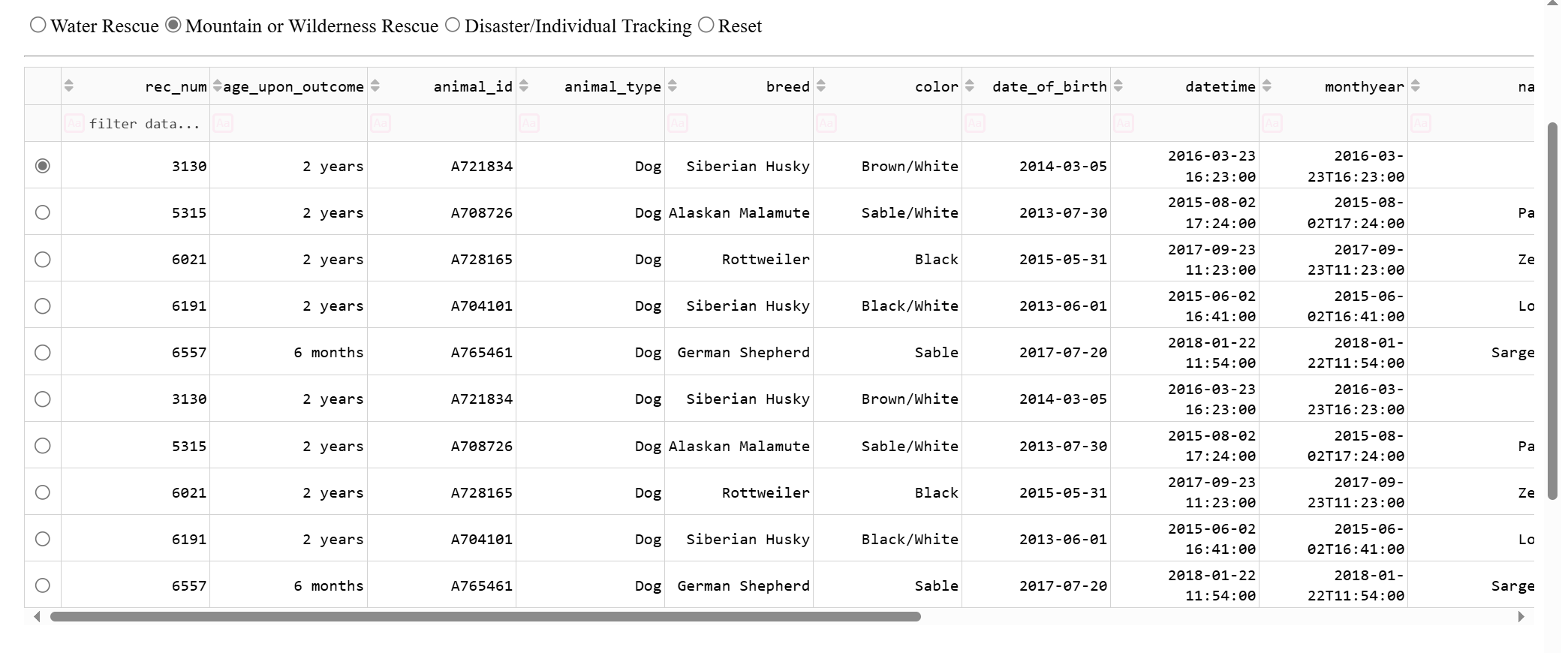
Lastly Jupyter Lab, Hosts and executes the notebook environment used for development and testing.

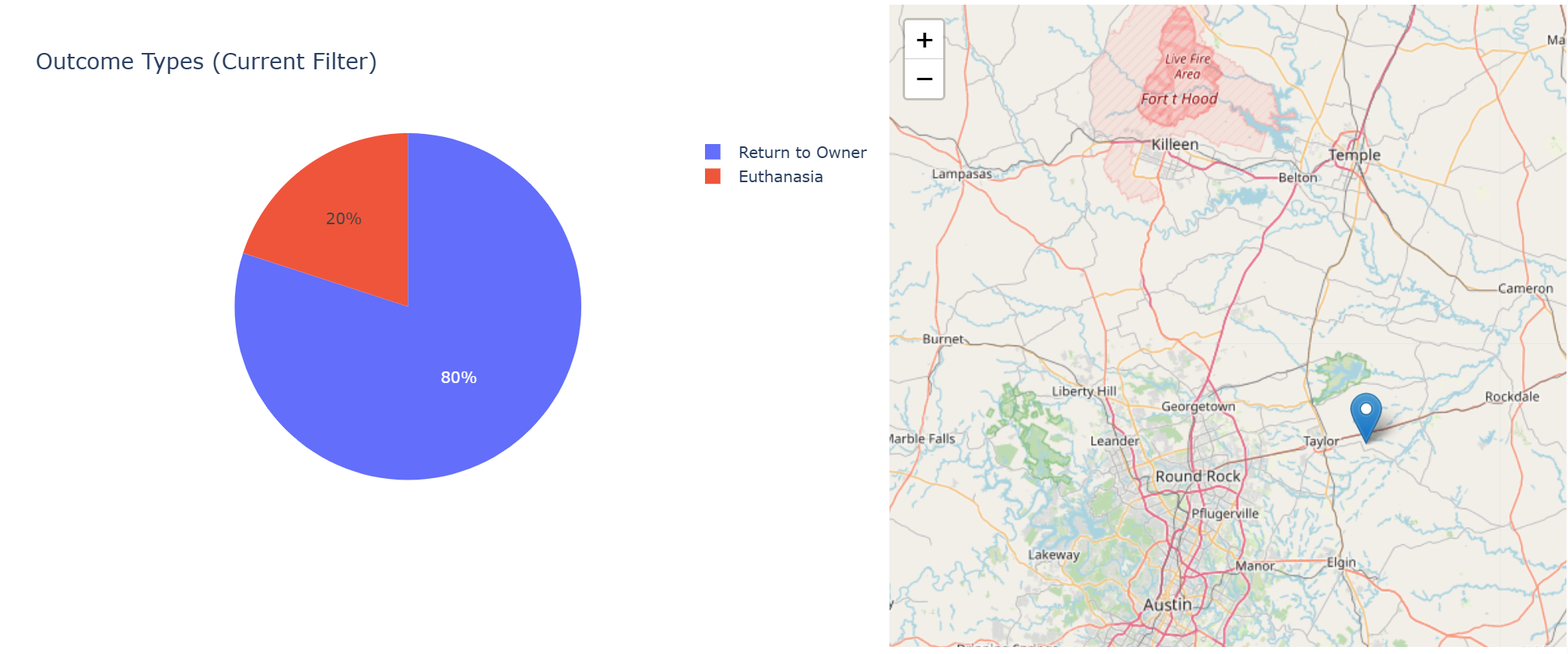
Here are the screenshots

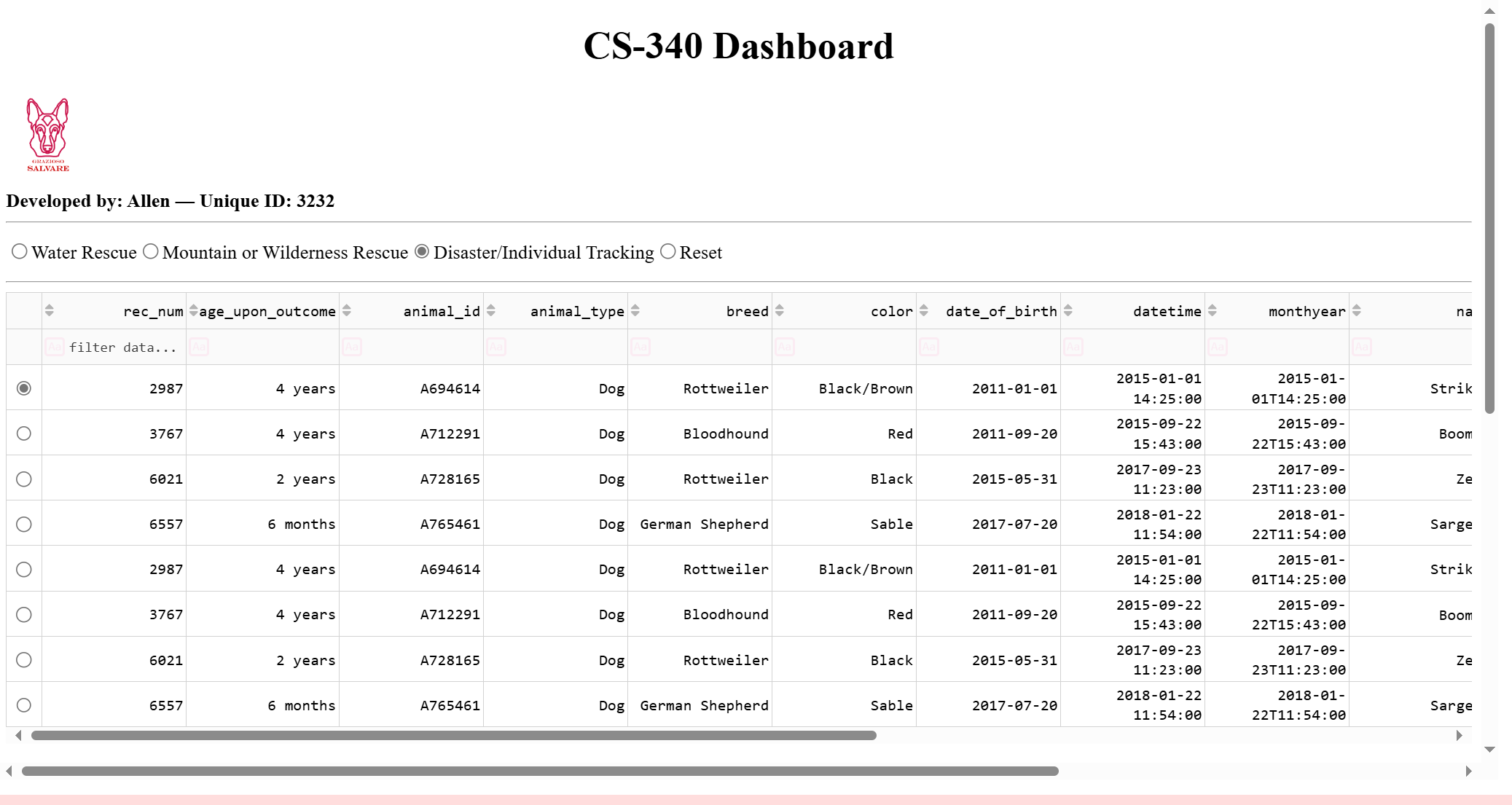
Starting Dashboard View  


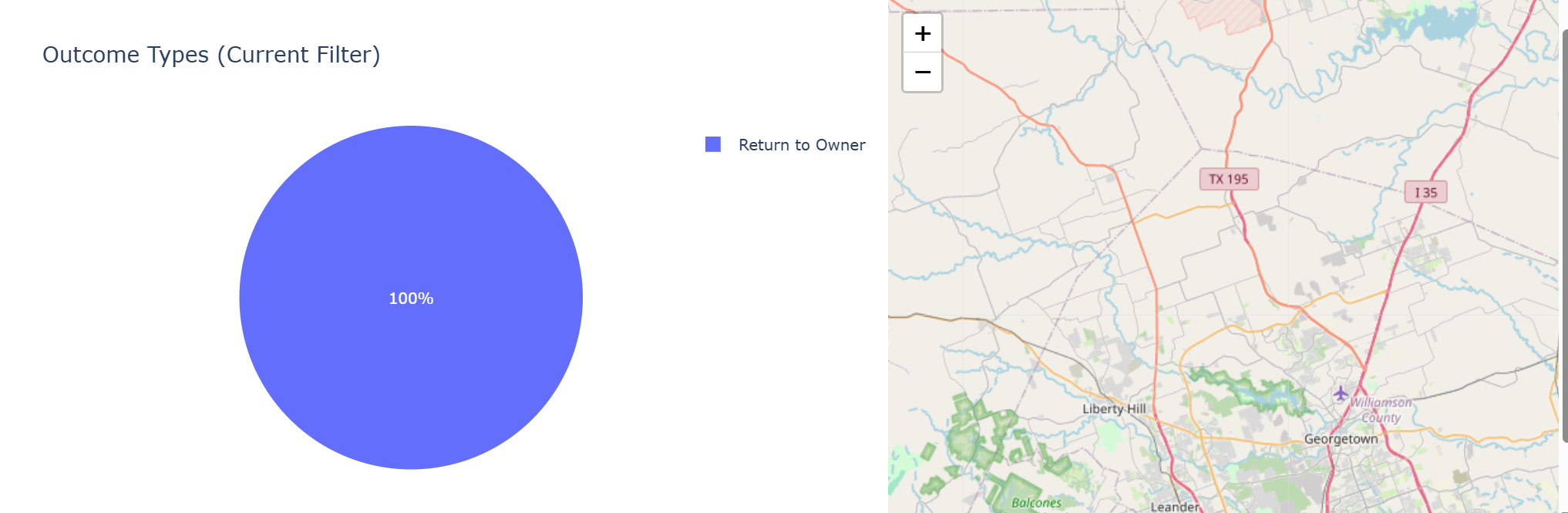
Water Rescue Filter  


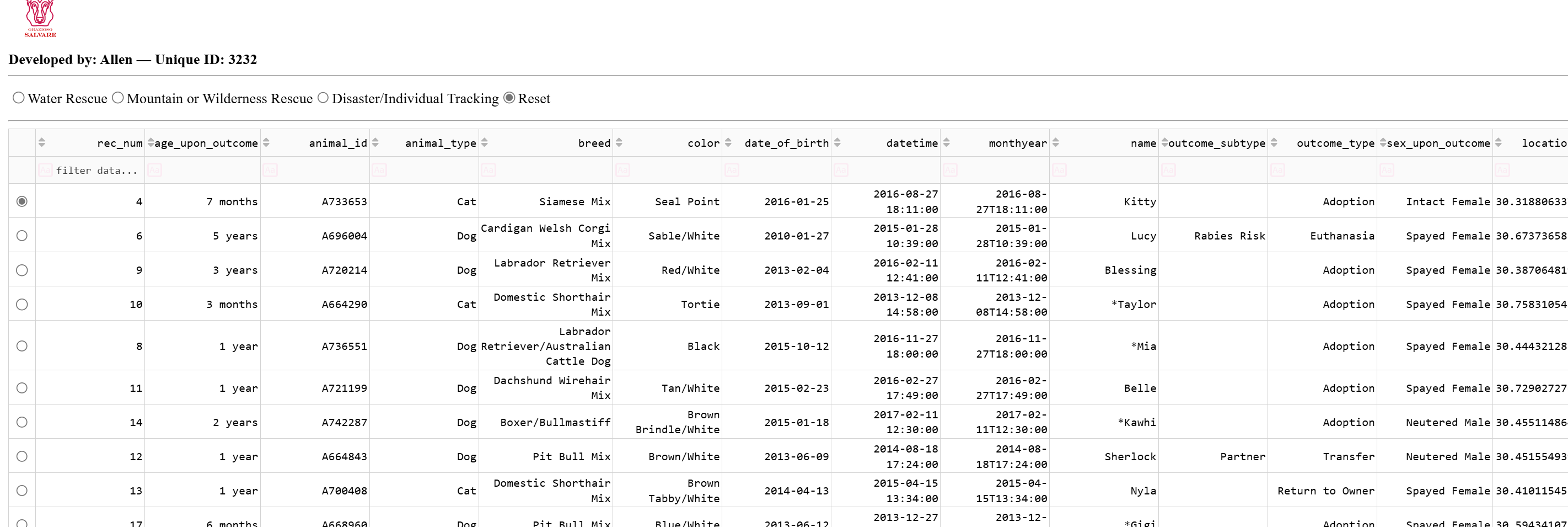


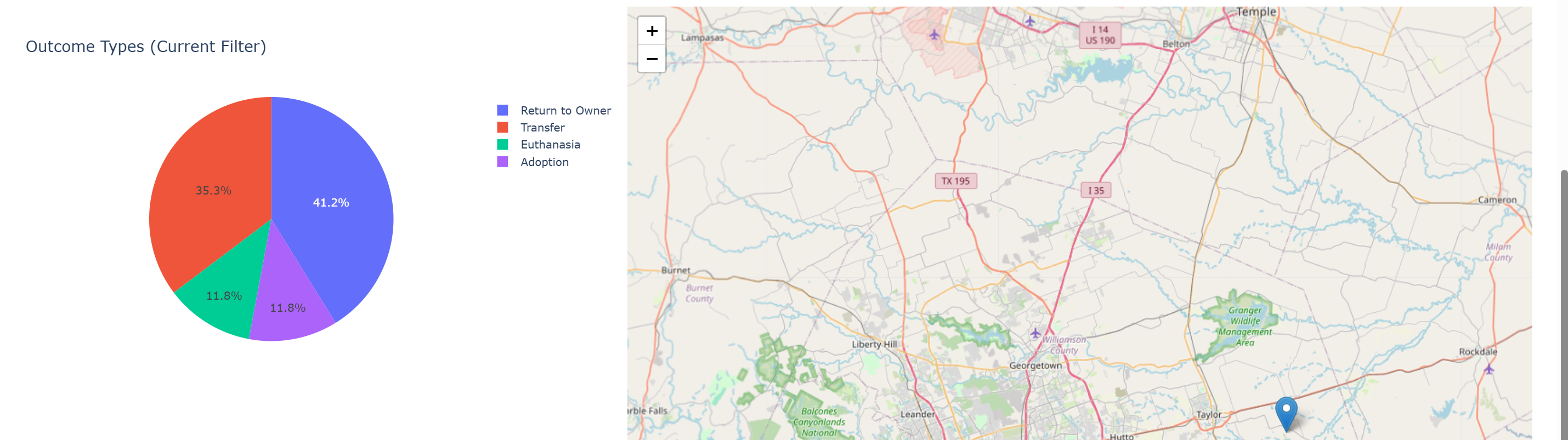
Mountain / Wilderness Rescue Filter  




Disaster / Individual Tracking Filter  




Reset Filter View  
 



Instructions  
Start MongoDB and Import Data

mongoimport --db aac --collection animals --type csv --headerline --file aac\_shelter\_outcomes.csv

Verify Database and Credentials

Username: aacuser Password: SNHU1234

Database: aac Collection: animals

Place Files in Same Directory (very important lol)

List of files

ProjectTwoDashboard.ipynb

CRUD\_Python\_Module.py

Grazioso Salvare Logo.png

Then after files

Open and Run Dashboard

Launch Jupyter Lab.

Open ProjectTwoDashboard.ipynb.

Run All Cells.

Dash will run on http://127.0.0.1:8051/ inside Jupyter.(hopefully)

After it opens use the Dashboard

Select any radio filter (Water / Mountain / Disaster / Reset).

Observe live updates in the data table, pie chart, and map.

Here are the steps taken to complete the project

So first Built CRUD Python Module to connect to MongoDB and support create, read, update, delete operations.

Then Imported data from the Austin Animal Center CSV into MongoDB.

This developed Dashboard using Dash layout with logo, radio filters, data table, map, and chart containers.

Some wired Callbacks:

update\_dashboard() → filters database queries.

update\_graphs() → renders live pie chart.

update\_map() → updates map marker for selected row.

Styled UI with pagination, sorting, and row highlighting.

Then tested each filter and validated that database data and visuals refresh correctly.

Challenge, Resolution

Had a 500 error when no column selected, Added a None check in update\_styles() callback.

Then had ObjectId crash, Dropped the \_id column in the DataFrame before display.

And had the port conflict with previous Dash apps, Changed port to 8051 in app.run\_server(mode='inline', port=8051).

Had a empty filters causing crashes, So Added guard statements (if dff.empty: return []) to handle no results safely.

Reflection

So developing this project reinforced how the MVC architecture separates logic cleanly:

Model (MongoDB) stores and serves data.

Controller (CRUD module) manages database operations.

View (Dash) provides an interactive client experience.

It also demonstrated how combining Python + Dash + MongoDB creates powerful, low-code full-stack applications.

References:

- Dash by Plotly Documentation – https://dash.plotly.com

- MongoDB Documentation – <https://www.mongodb.com/docs/>

-SNHU course information