# Grazioso Salvare: README

Riley Eichenour

Southern New Hampshire University

CS340: Client/Server Development

Professor Rowan Liwanag

October 16, 2025

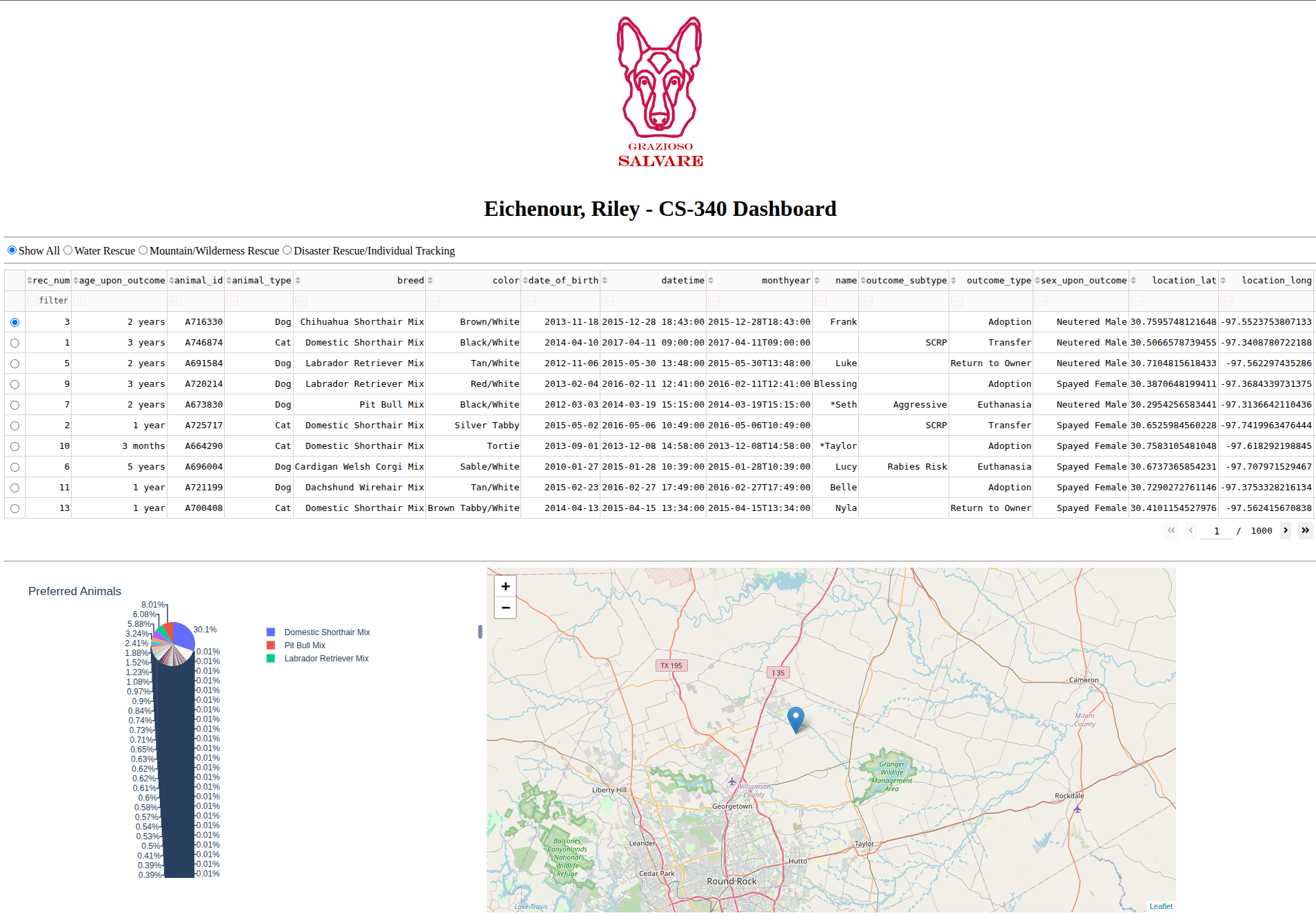
# Grazioso Salvare: README

### Functionality Requirements

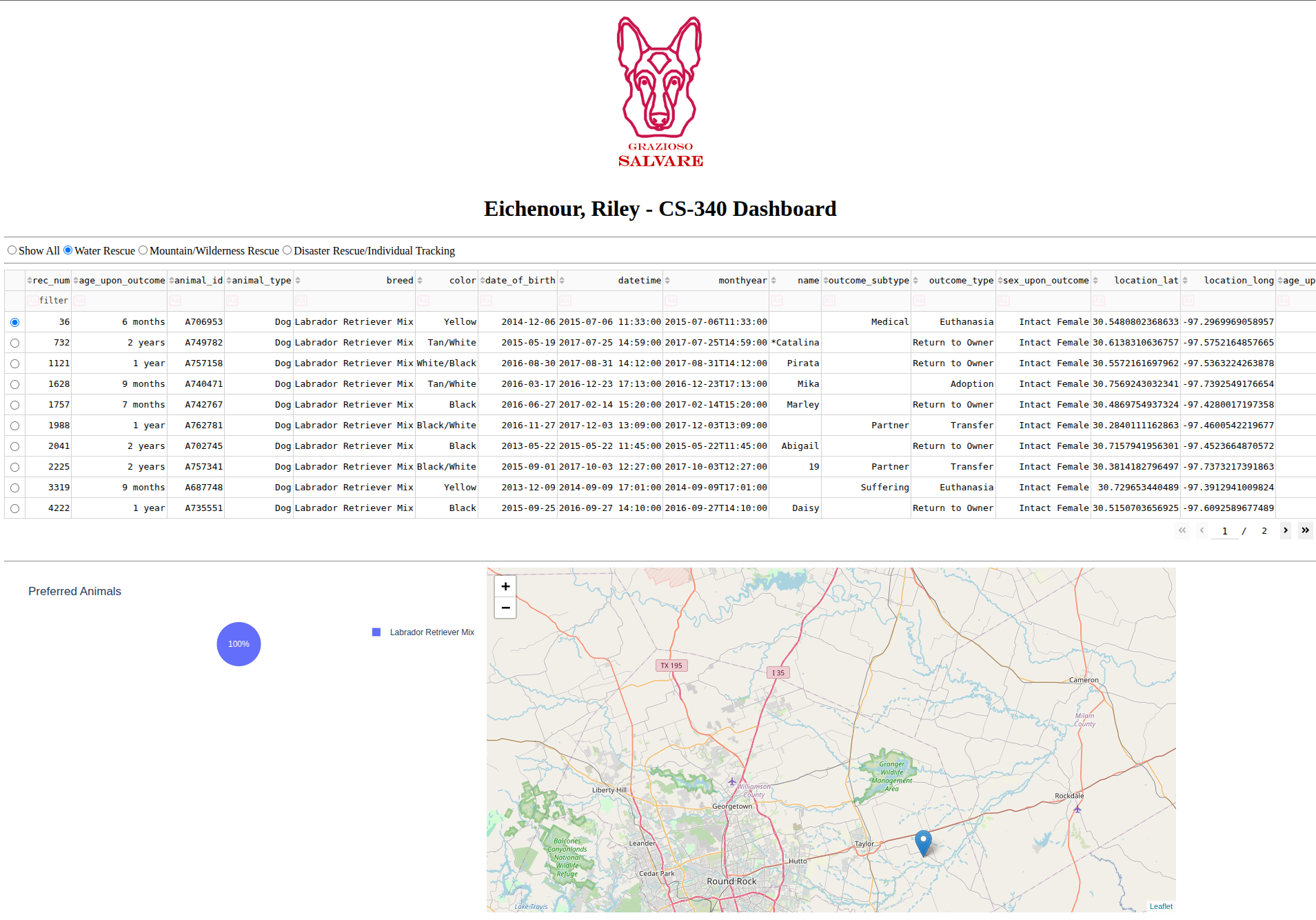
#### Interactive Filtering

Users are able to filter the data visible in the table with the following filters:

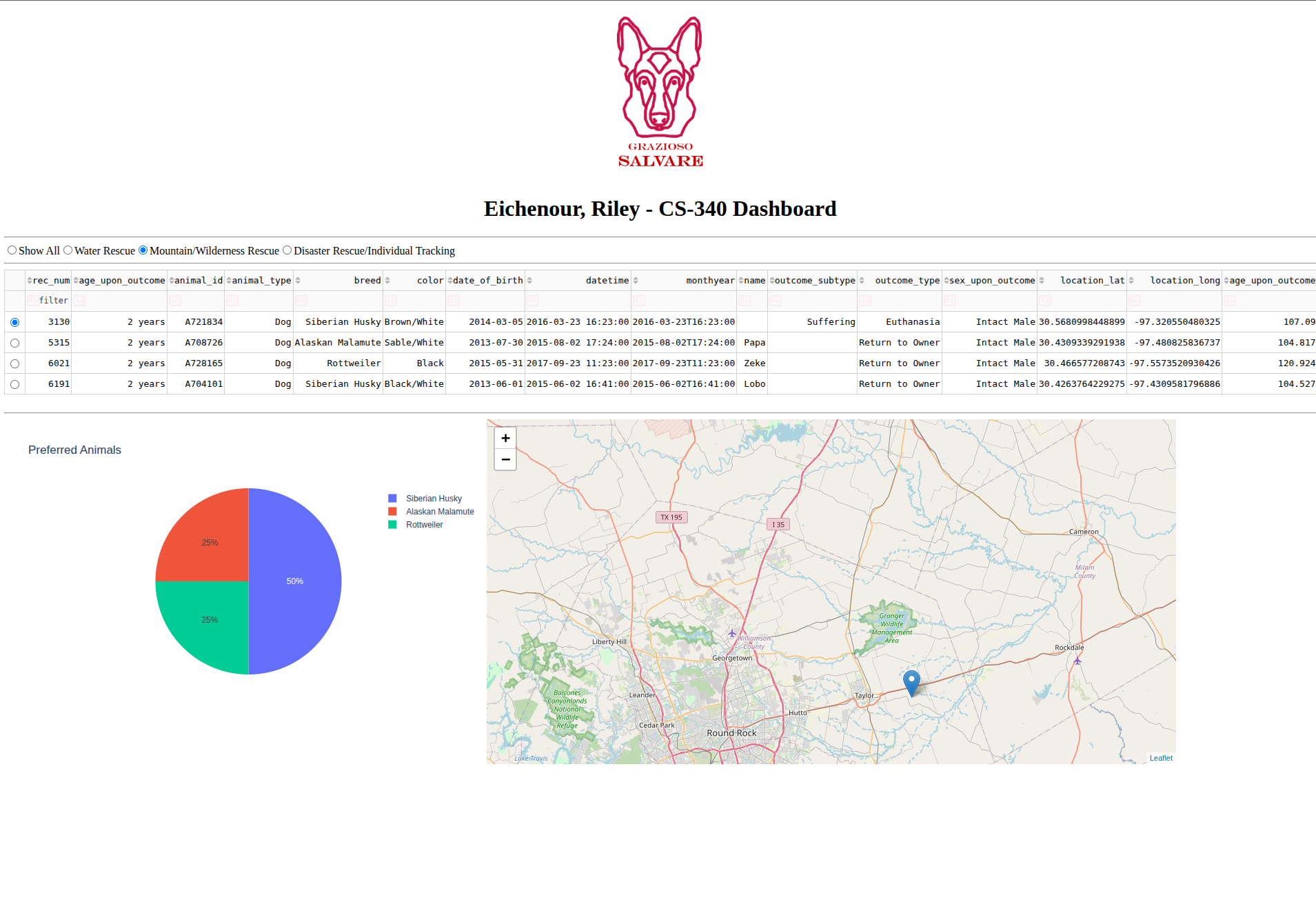
* + - Show All



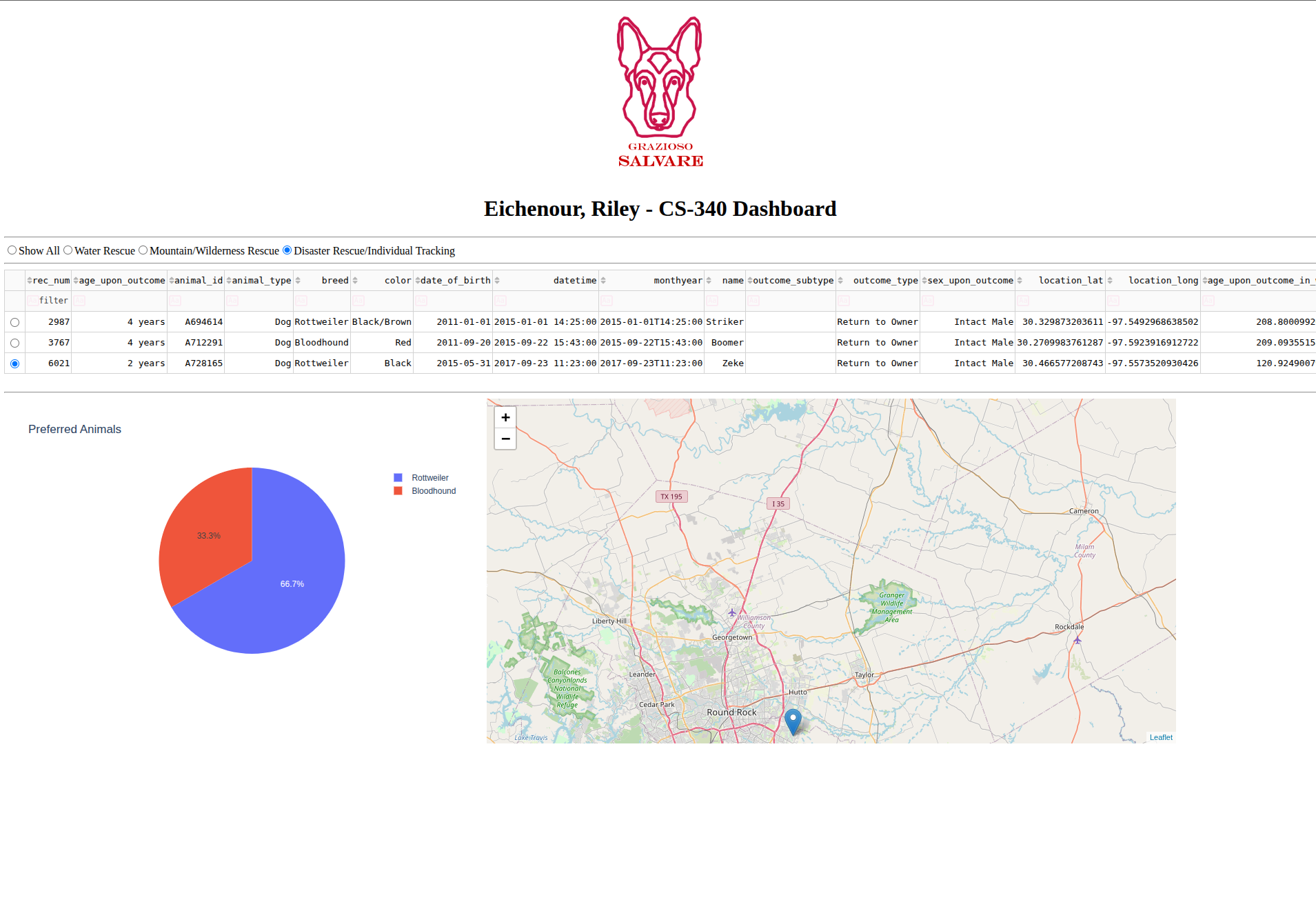
* + - Water Rescue



* + - Mountain/Wilderness Rescue

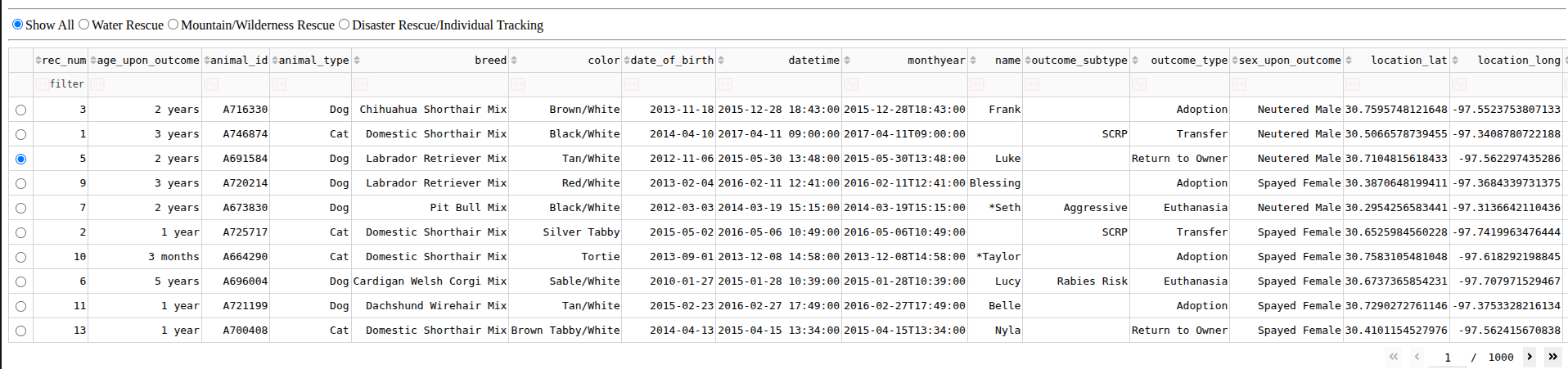


* + - Disaster Rescue/Individual Tracking



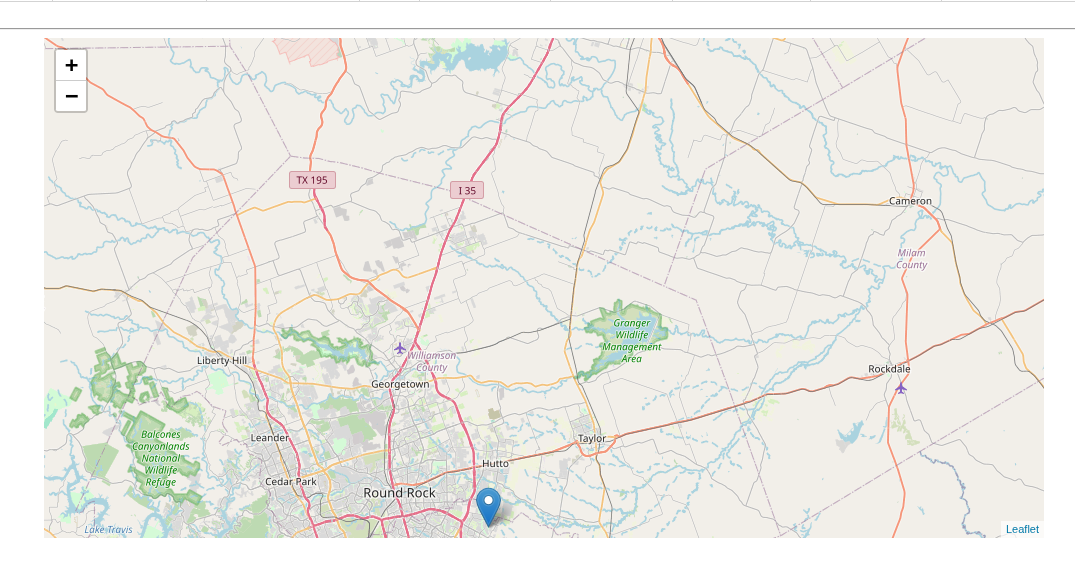
#### Data Table

The information is presented in an eassily viewable table format that shows the information of each animal. The applied filters will control what information is visible.



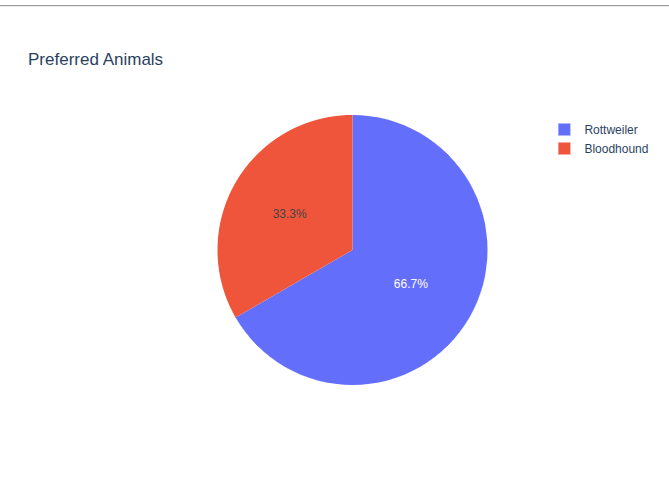
#### Geolocation Chart

A geolocation chart is visible that will show the location of the selected animal on a map.



#### Pie Chart

Pie chart that shows the breed distribution of the currently filtered animals.



### Application Technologies

#### MongoDB

The data persistence layer for the application was based on MongoDB, which is a NoSQL database with a document model. The NoSQL schema was a valid option for this project to account for storing different structures in the animal documents as the application grows or the needs shift. It also provides built in authentication control to easily control who has access to the database. Coupled with PyMongo is makes data retrieval, manipulation, and filtering quick through frameworks like Pandas. This allows developer’s and data analysts to be able to work on the application.

#### Dash Framework

With the MongoDB and PyMongo driven backend it was important to aim to keep the language across the stack consistent. This layer handles the view and controller portions of the application to decouple the way the information is manipulated in the business logic and the data layer. The quick integrations with Pymongo, Pandas, and Dash made it easy to get an operational panel for user utilization and quick time to utilization.

### Project Setup

This application has the following dependencies:

* + - PyMongo – 4.13.X - <https://pymongo.readthedocs.io/en/stable/index.html>
    - MongoDB – 7.0.X - <https://www.mongodb.com/>
    - Python – 3.11.2 - <https://www.python.org/downloads/release/python-3110/>

It is recommended to install the dependencies through a Python environment manager, such as pip, for PyMongo and Python. Refer to <https://www.mongodb.com/docs/manual/installation/> for more information on setting up MongoDB. Refer to <https://pip.pypa.io/en/stable/installation/> for an example of a Python environment manager.

The following Python dependencies need to be installed:

* + - Plotly - https://plotly.com/python/
    - Dash Framework - https://dash.plotly.com/
    - numpy - https://numpy.org/
    - pandas - https://pandas.pydata.org/
    - matplotlib - https://matplotlib.org/

If data is available utilize Mongosh to import the data into a database named AAC with an animals collection. Once the data has been imported and the MongoDB service is accessible start the application script. In a web browser navigate to the URL output to view the Dashboard.

### Challenges

One challenge I encountered was being unfamiliar with debugging the Dash Framework and had an issue related to the returns from one of my callback functions not being in the expected format following the commented out code in the application. I was able to find more examples in the Dash Framework documentation that provided enough information to isolate what part of my callback was being problematic despite the cryptic errors produced in the browser as a result of the code.