**Dashboard Readme**

****

## About the Project/Project Title

The Austin Animal Center is currently upgrading their animal database systems to not only improve functionality for their employees but to also improve the reliability of keeping cohesive data on all animals currently being taken care of. This project is a script that allows an employee of the Animal Center to quickly and easily read, add, remove, or update entries in the database with little knowledge of the underlying database code.

## Motivation

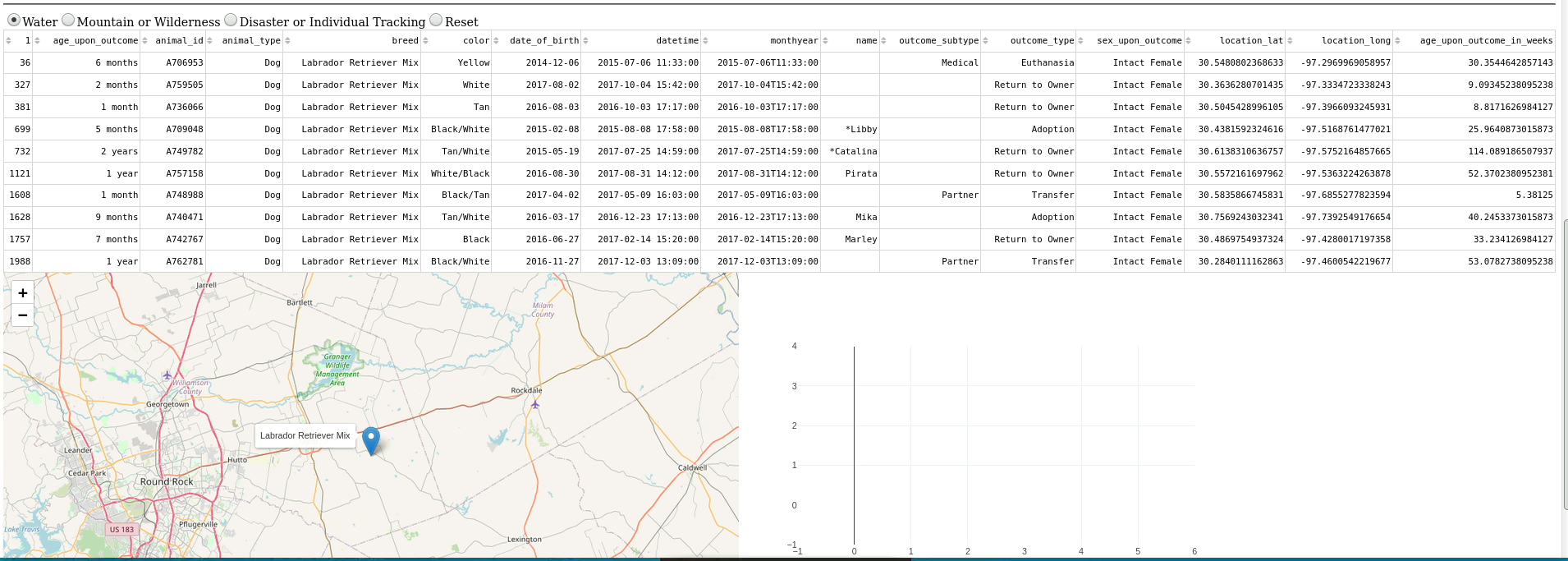
The Austin Animal Center is a paramount institution that serves the people and pets of the Austin Texas area and its employees should have quick and easy access to their database functionality at a moments notice without having to be sent to additional training courses before hire.

**Functionality**

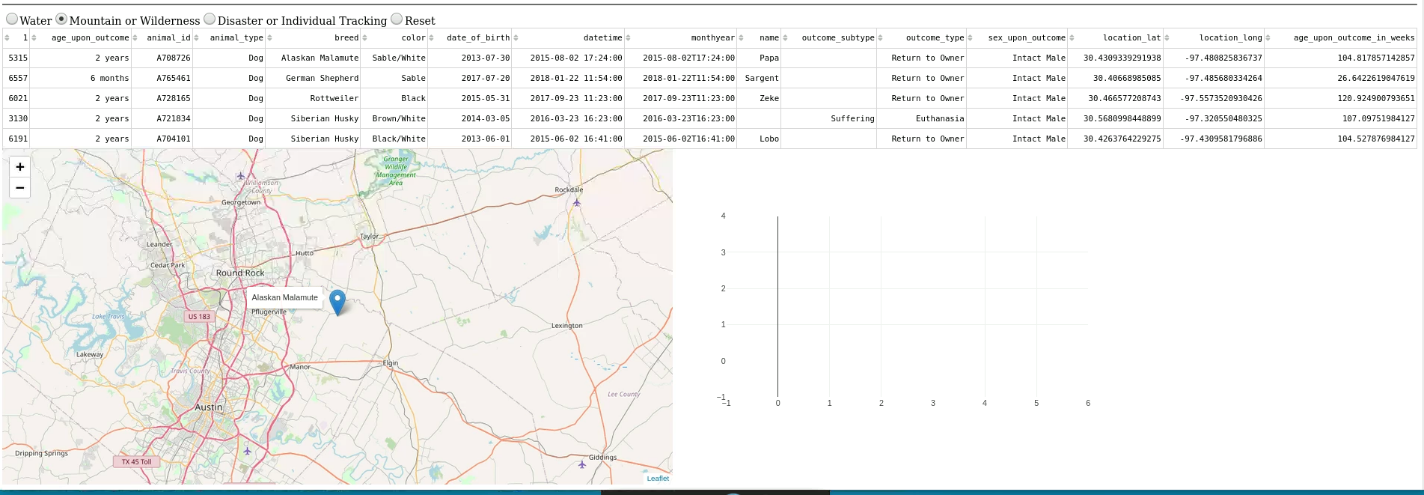
As shown in the figures below, the dashboard for the database allows the user to select 4 pre-set filters to categorize data at the users’ whim by selecting the radio buttons at the top of the data table.

\*I unfortunately couldn’t get the Pie chart to display properly…this is as far as I could manage.

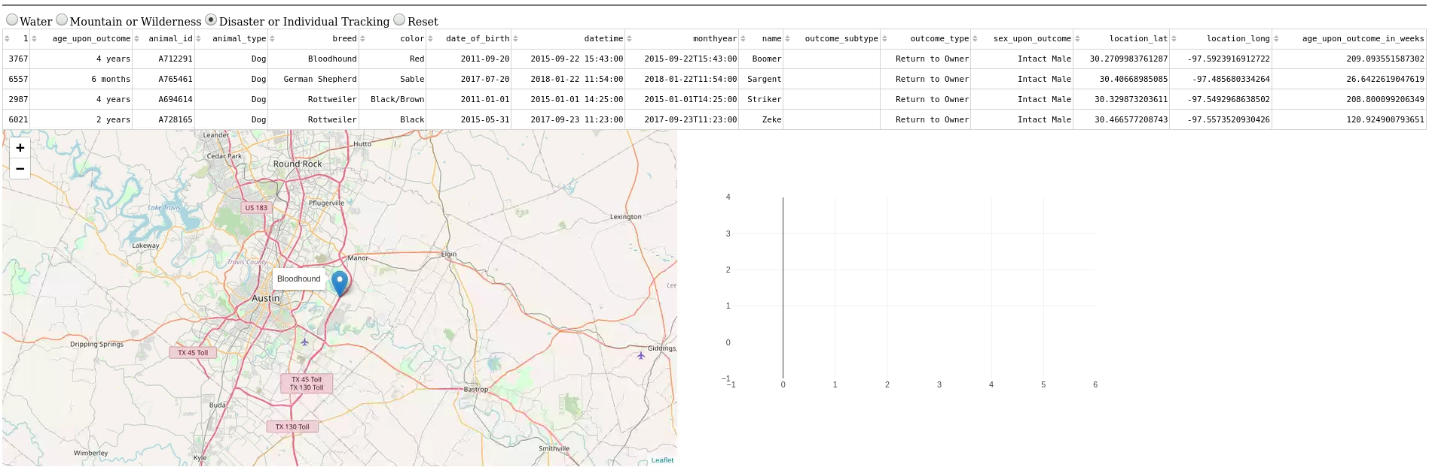
**Water Rescue:**



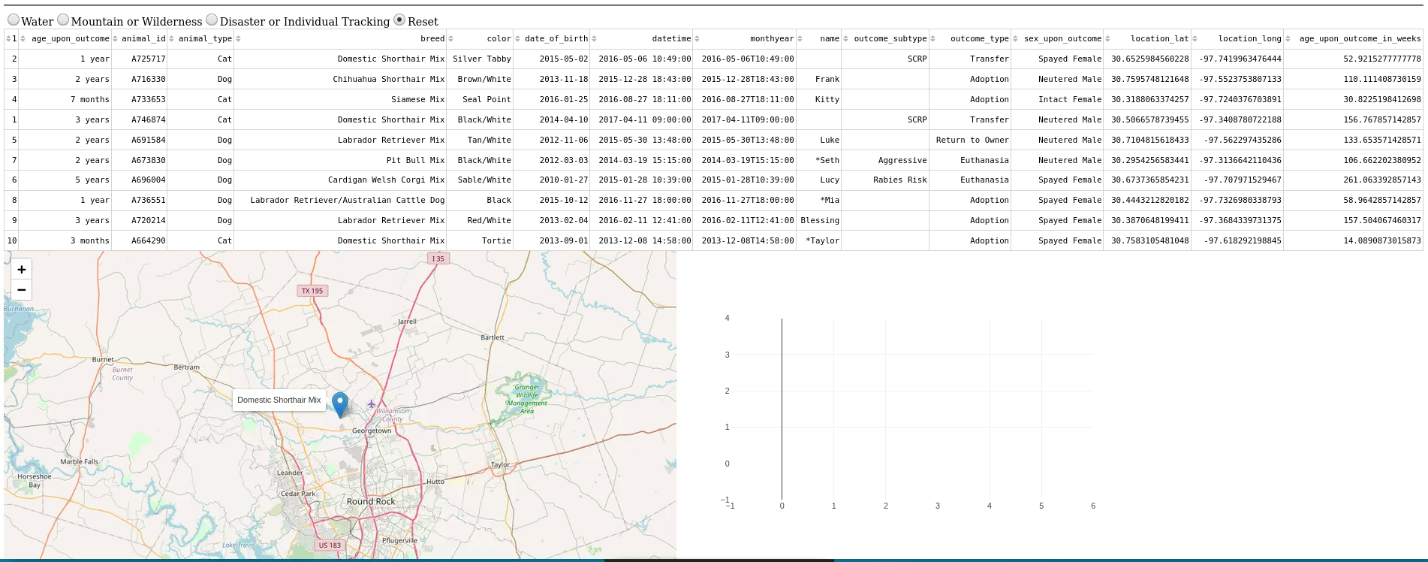
**Mountain or Wilderness Rescue:**



**Disaster Rescue or Individual Tracking:**



**Reset: (Resets filter to default values)**



**Tools**

MongoDB was used for the underlying database of this dashboard. Its robust functionality, along with its synchronicity with Python, allowed for a simple yet well-rounded display of data. PyMongo, the python distribution that was used in building the dashboard, made writing scripts that augmented the MongoDB database in various ways very intuitive, and the Dash framework brought it all together in an elegant display.

**Steps**

After setting up the Mongo database via the Linux terminal and the python script to modify it, as described in the AAC Readme, the next natural step was to display all of our data. Using the Dash framework I displayed the data in a dashTable and added filters for the various rescue groups that were mapped to the 4 radio buttons located at the top of the table. The location map simply pulls the location\_lat & location\_long from the table and displays the location on the map along with a pin that displays the name and breed of the dog.

**Challenges**

The only real challenge I faced was getting the Pie Chart to display. As seen above, I couldn’t solve this before my allotted time expired. Displaying these axes (despite the code commanding a pie chart), was the closest I could get to a solution. I tried a number of fixes, none of which worked. That my issue could be rooted in one of two areas:

1. I suspect the ‘values’ variable when initializing the chart requires input that is specifically a percentage of 100, which is a value type I don’t currently have in the database. So I would have to calculate the percentage of each different breed by counting them in the dataFrame and dividing by the total. Then passing *THAT* value in may have worked.
2. My second thought is that the @app.callback information is fundamentally messed up. I tried plotting with the standard pandas dataFrame pie plot and that didn’t work either so that led me to believe that the callback was faulty. Whether I’m using the wrong inputs and outputs etc.