# CS-340: Project Two README

## About the Project/Project Title

The purpose of this project is to create a Python web application dashboard to make the viewing of data from a MongoDB database easier using Dash and a Python “middle-ware” interface. The interface streamlines CRUD operations between the MongoDB database and the Python dashboard.

## Motivation

The project facilitates the ease of viewing data by using a Python dashboard.

## Getting Started

The first real step to getting the project to work is setting the appropriate credentials in the \_\_init\_\_ method of the class and that the database you want to manipulate is imported properly. The database utilizes authentication thus the requirement for credentials. After that, you need to make sure that the “middle-ware” interface is located in the same directory as the Python dashboard. Then, it’s just a matter of setting up the dashboard using Python and the Dash framework. The database used in the examples is a database of animals from the Austin Animal Center (AAC).

## Installation

MongoDB: The type of database used in this project because it’s schema-less and can easily handle the necessary data

Python: The programming language used to interface with the MongoDB database

PyMongo: The driver used to connect to and manipulate the MongoDB database with Python code

Jupyter Notebook: The IDE used to create the class and the script used to test it

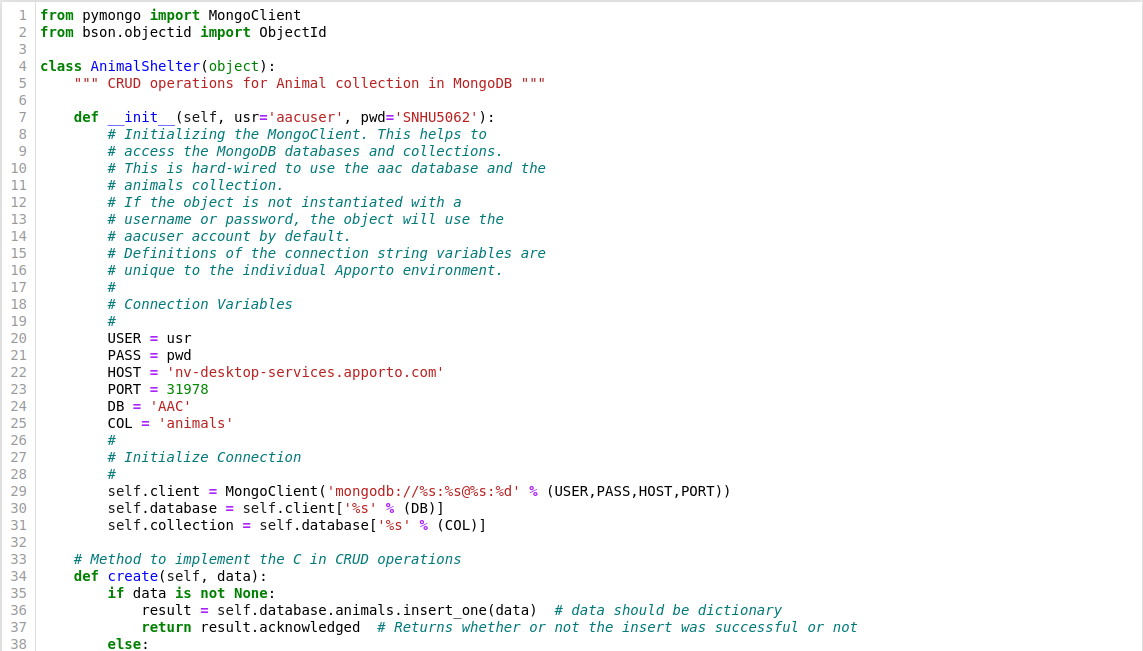
Dash: The framework used to create the Python dashboard

## Usage

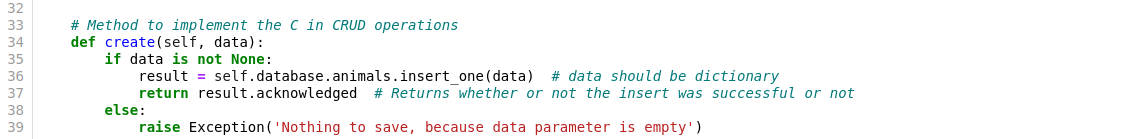
This project allows the user to easily view, sort, filter, and analyze the data in the database you connect it to.

### Code Example

Connection:



This is the code used to connect the interface to the desired MongoDB database. In this case it’s connecting to a database of animals from the Austin Animal Center.

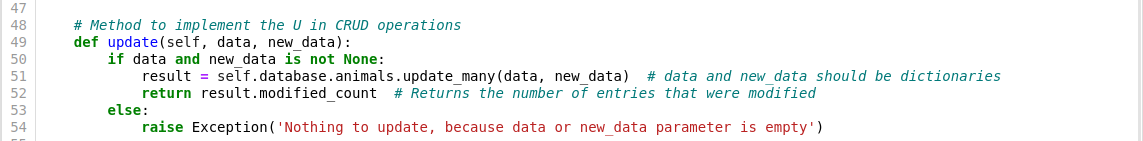
Create:

This is the interface code used to insert a provided parameter into the connected database. In this case the data needs to be in the form of a dictionary to ensure proper insertion.

Read:

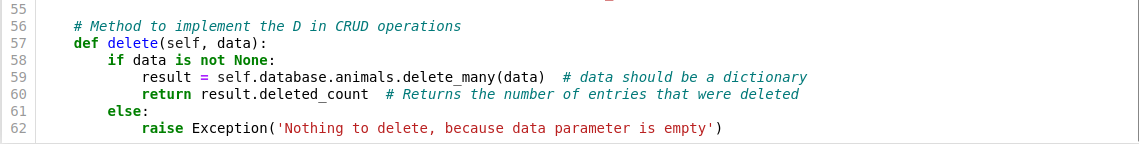


This is the interface code used to search for documents in the connected database using a key-value pair for search terms. It then returns the resulting documents in a list.

Update:

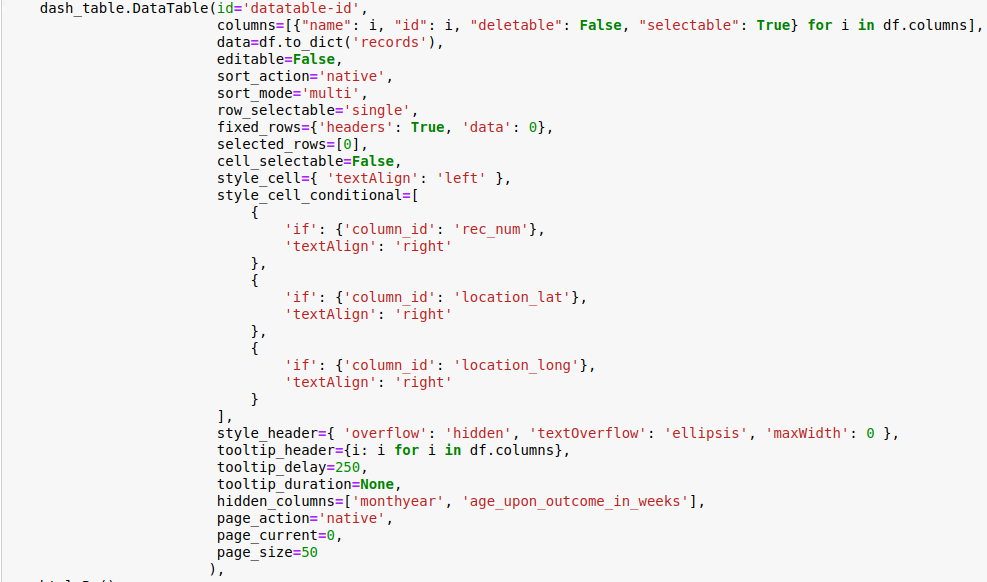
This is the interface code used to update entries in the database. It searches the database for the entries that match the first key-value pair and replaces the values specified in the second parameter of key-value pairs. It returns the number of entries that were modified by that operation.

Delete:



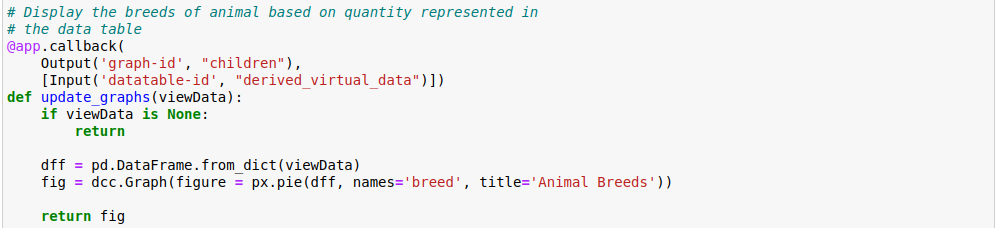
This is the interface code used to delete existing entries in the database. It searches for entries that match the provided key-value pairs and then removes them from the database. It returns the number of entries that were deleted in that operation.

Display:



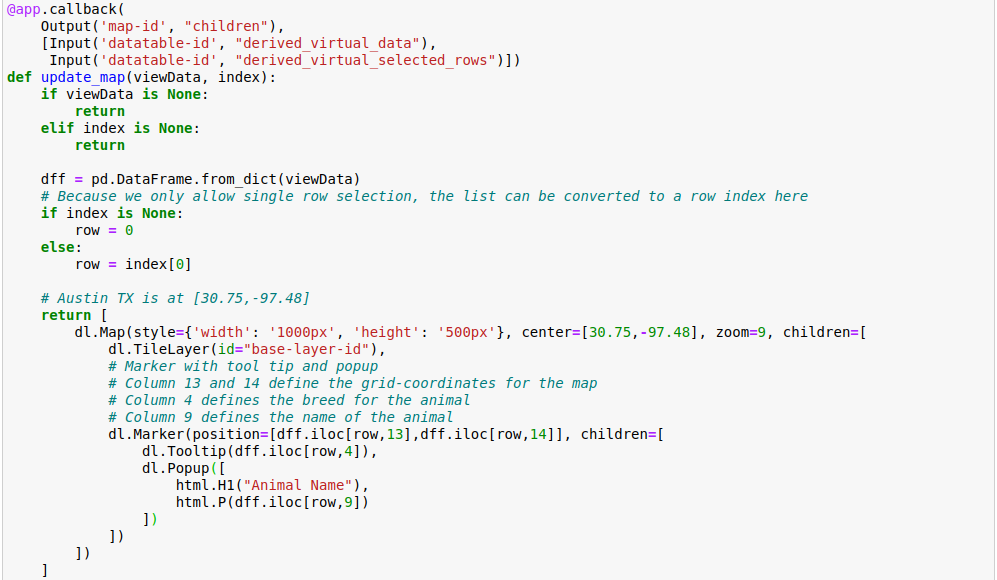
This is the Python web application code used to display the data from the MongoDB database in an easy to view manner. It allows the user to sort and filter the data and choose how it’s viewed.

Graph:



This is the Python web application code used to display the graph of dog breeds in the viewed database.

Map:



This is the Python web application code used to display the geolocation element in the dashboard to view where the animals are located.

### Tests

The best way to run tests is by importing the project, instantiating a class object and trying out each of the methods.

Code for creating an entry:

# Instantiate the AnimalShelter class with the proper user credentials

shelter = AnimalShelter('aacuser', 'SNHU5062')

# Create the data to insert into the animal collection

data = {'rec\_num': 10001,

'age\_upon\_outcome': '1 year',

'animal\_id': 'A856985',

'animal\_type': 'Dog',

'breed': 'Cardigan Welsh Corgi',

'color': 'Tan/White',

'date\_of\_birth': '2022-04-15',

'datetime': '2023-05-26 17:52:00',

'monthyear': '2023-05-26T17:52:00',

'name': '\*Caesar',

'outcome\_subtype': '',

'outcome\_type': 'Return to Owner',

'sex\_upon\_outcome': 'Intact Male',

'location\_lat': 30.5237503840071,

'location\_long': -97.4065794969578,

'age\_upon\_outcome\_in\_weeks': 58

}

# Test the create(data) method

create\_test = shelter.create(data)

print('Create Test Result:', create\_test)

The code returns:

Create Test Result: True

The code to read the just entered information:

# Test the read(data) method

read\_test = shelter.read({'animal\_id': 'A856985'})

print('Read Test Result:', read\_test)

The code returns:

Read Test Result: [{'\_id': ObjectId('647a99f1caccb54384835339'), 'rec\_num': 10001, 'age\_upon\_outcome': '1 year', 'animal\_id': 'A856985', 'animal\_type': 'Dog', 'breed': 'Cardigan Welsh Corgi', 'color': 'Tan/White', 'date\_of\_birth': '2022-04-15', 'datetime': '2023-05-26 17:52:00', 'monthyear': '2023-05-26T17:52:00', 'name': '\*Caesar', 'outcome\_subtype': '', 'outcome\_type': 'Return to Owner', 'sex\_upon\_outcome': 'Intact Male', 'location\_lat': 30.5237503840071, 'location\_long': -97.4065794969578, 'age\_upon\_outcome\_in\_weeks': 58}]

The code to update the entered information with an updated birth date:

# Test the update(data, new\_data) method

update\_test1 = shelter.update({'name': '\*Caesar'}, {'$set': {'date\_of\_birth': '2022-03-15'}})

print('Update Test Result:', update\_test, 'entries updated')

The code returns:

Update Test #1 Result: 1 entries updated

The code to delete the created entry:

# Test the delete(data) method

delete\_test = shelter.delete({'rec\_num': 10001})

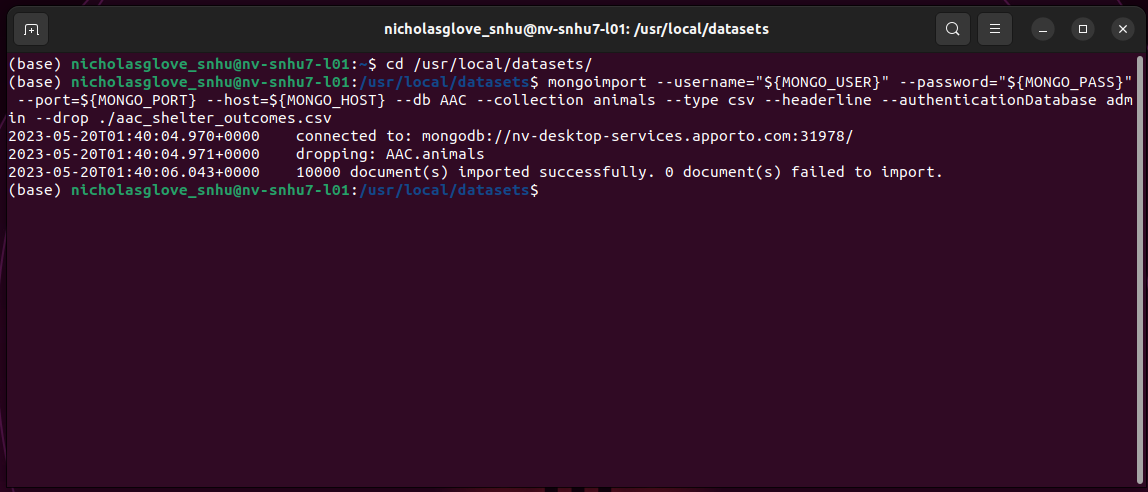
print('Delete Test Result:', delete\_test, 'entries deleted')

The code returns:

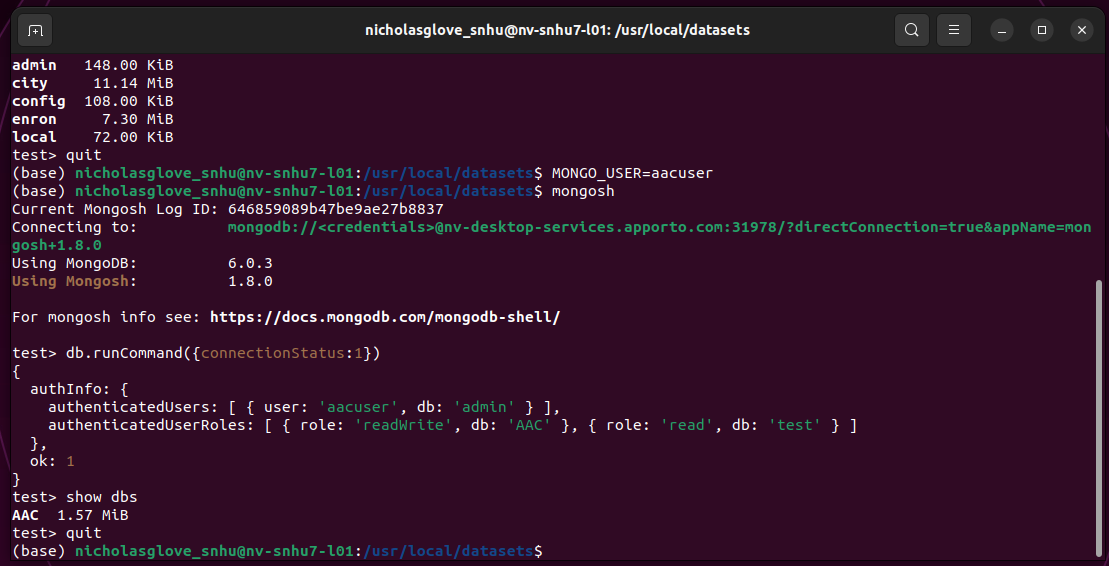
Delete Test #1 Result: 1 entries deleted

### Screenshots

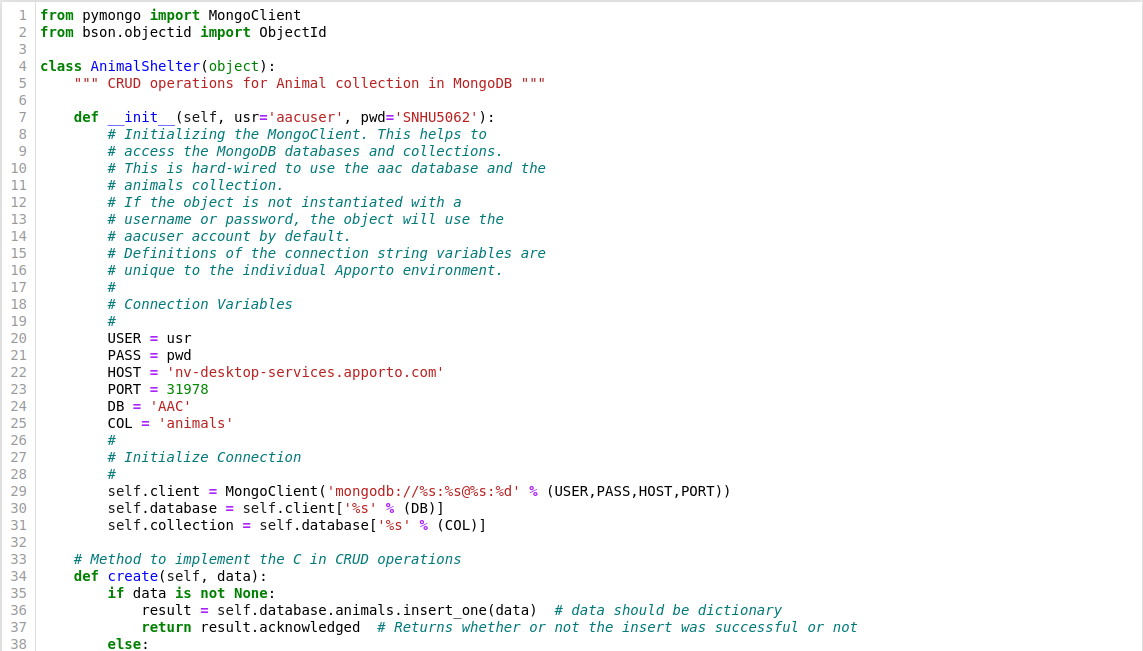
Screenshot of the MongoDB database import:

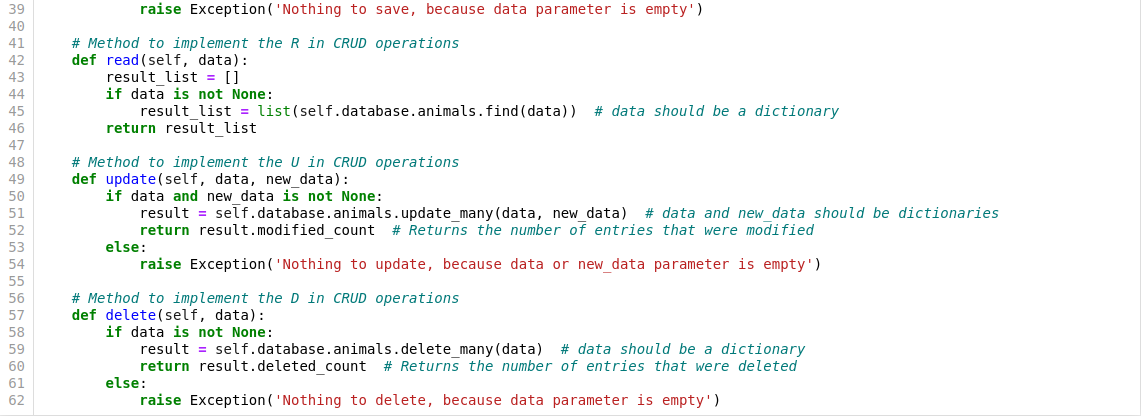


Screenshot of the user authentication of the database:

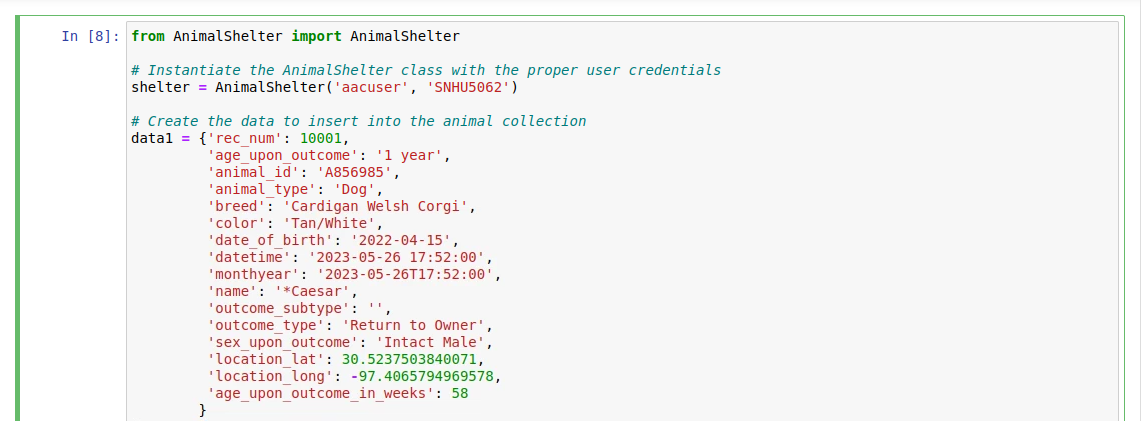


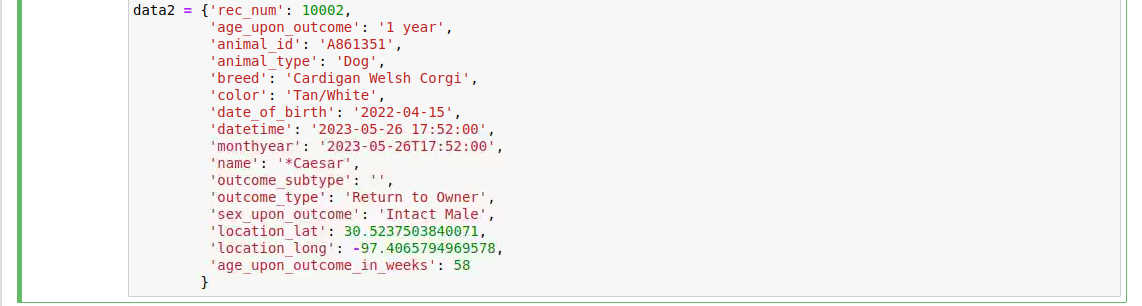
Screenshots of the code for the class object:

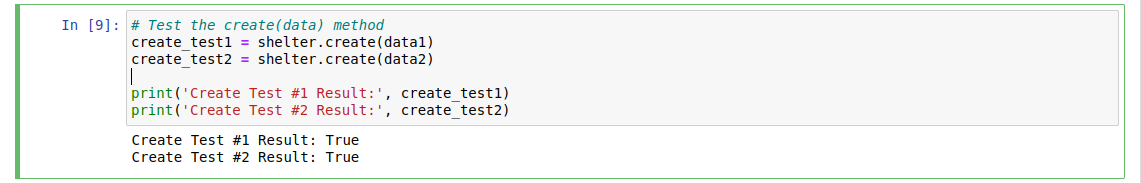




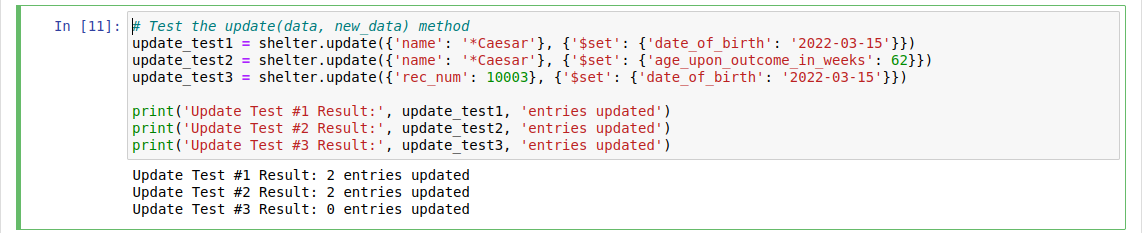
Screenshots of the code for the test script:

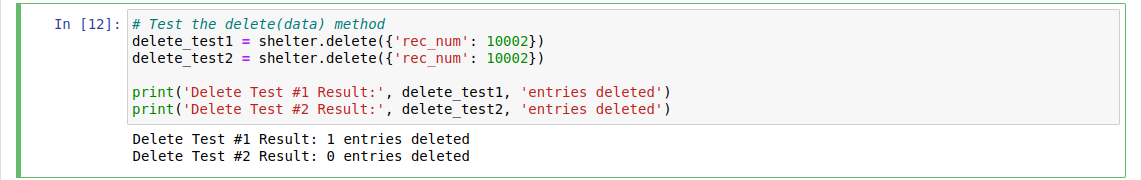




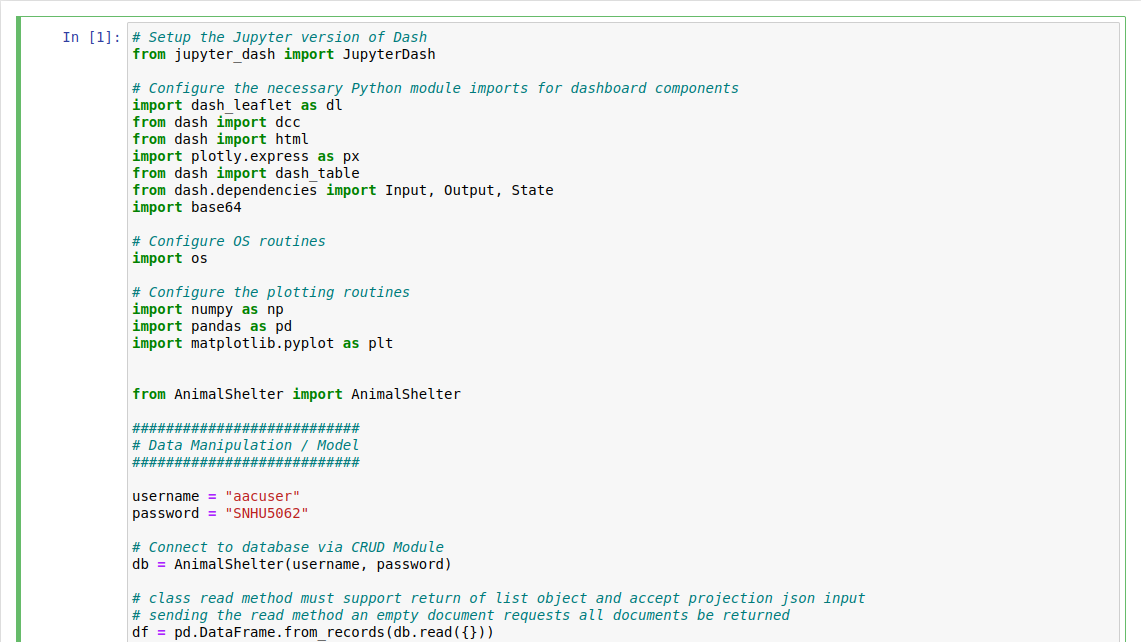


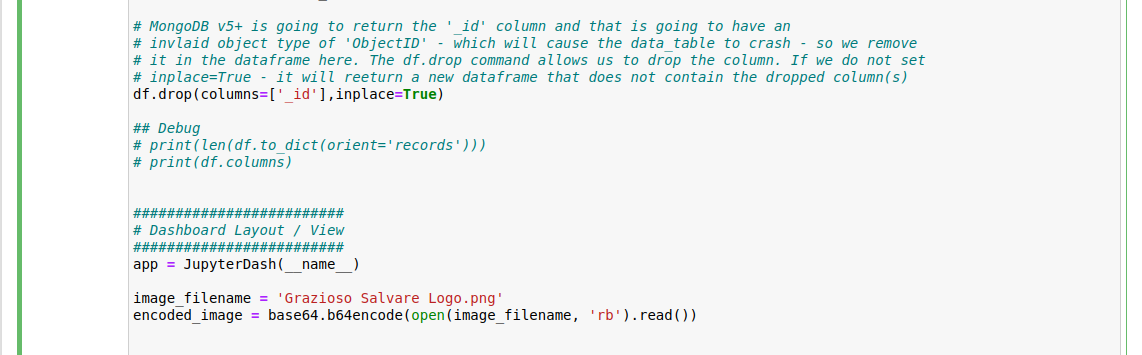






Screenshots of the dashboard code:



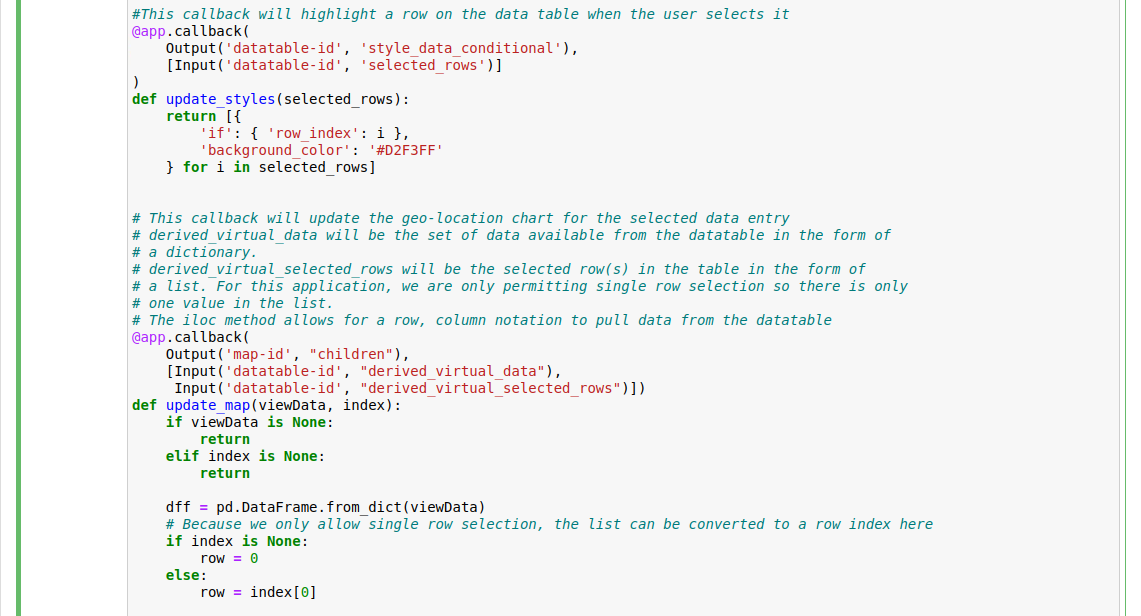


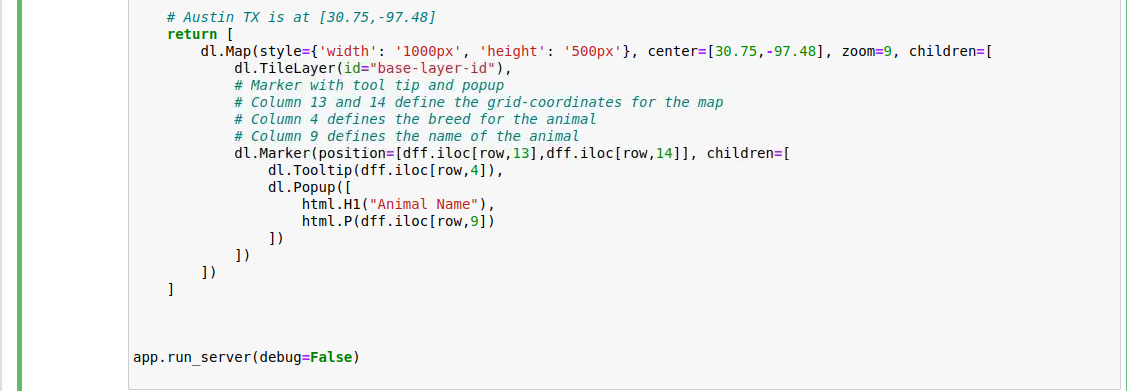








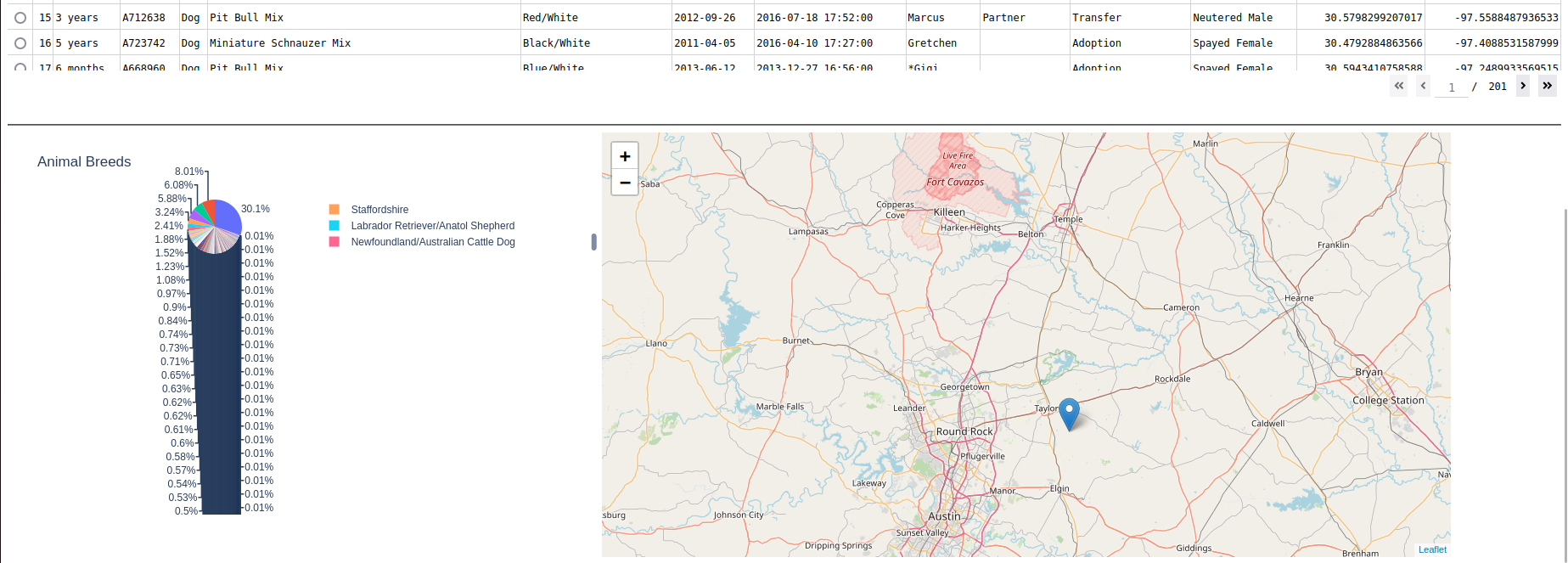




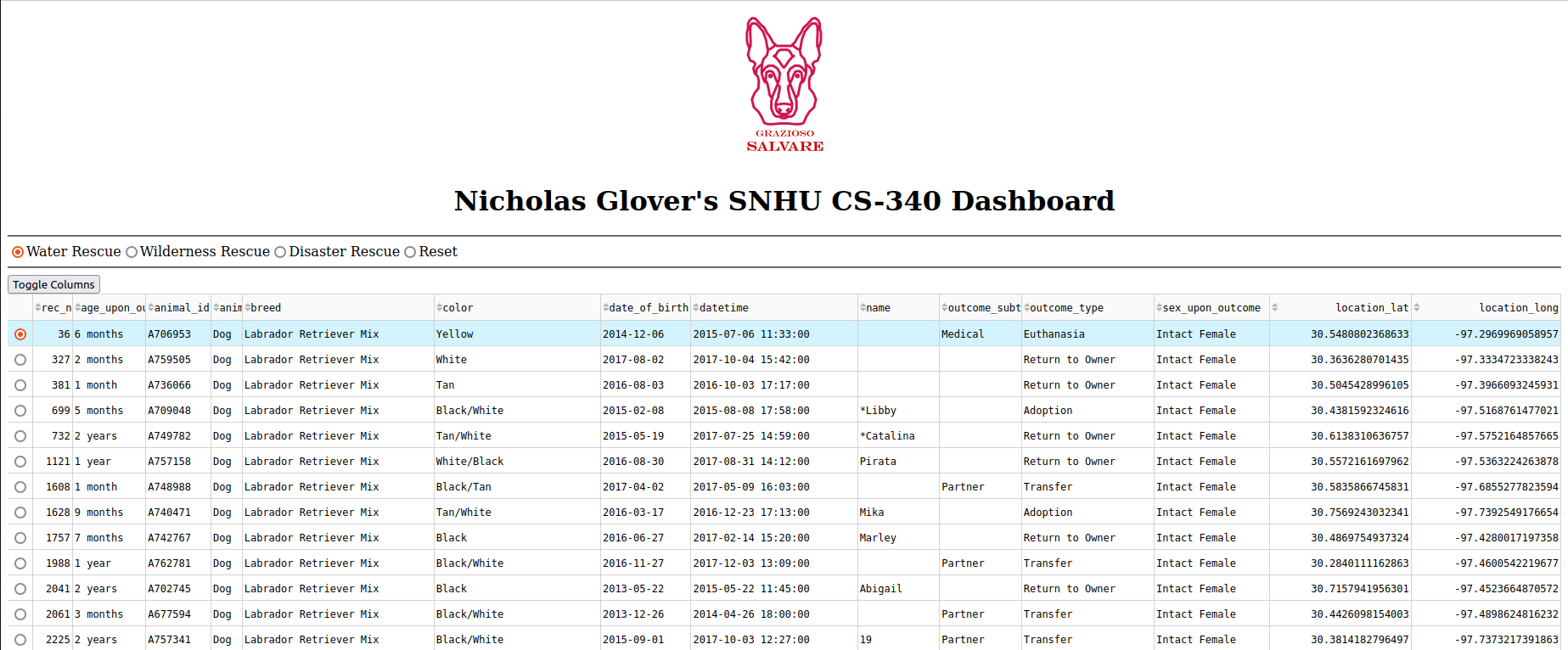
Screenshots of the dashboard:

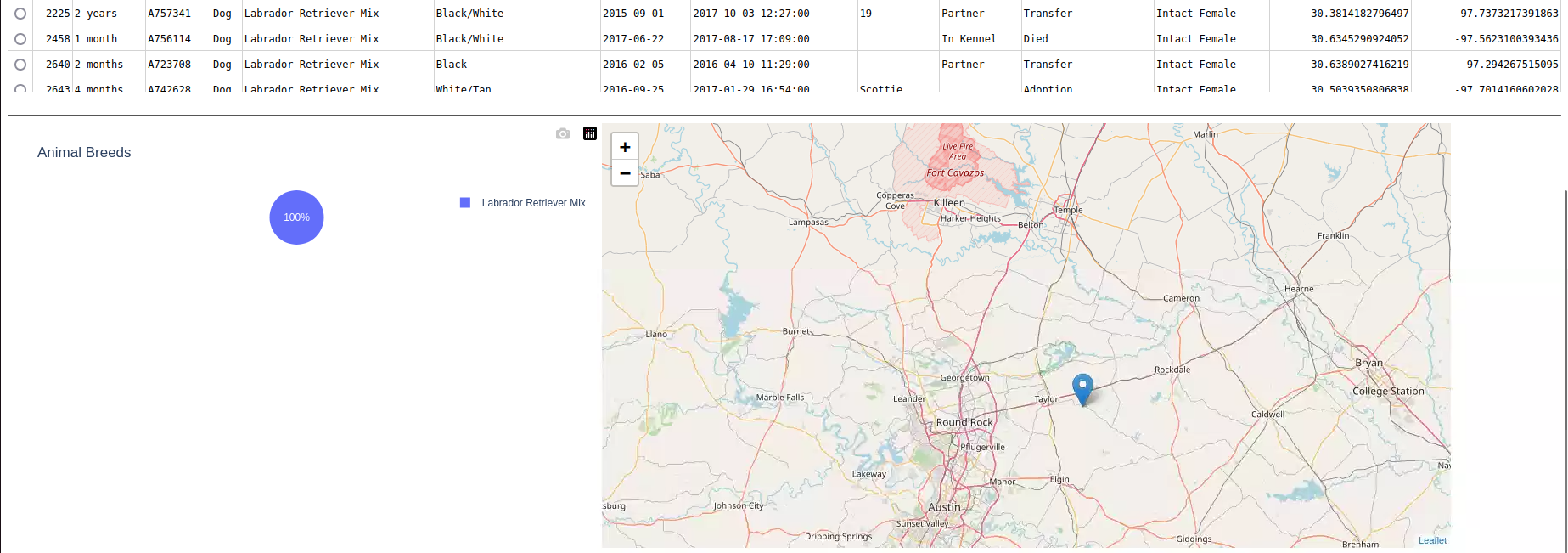
Unfiltered:



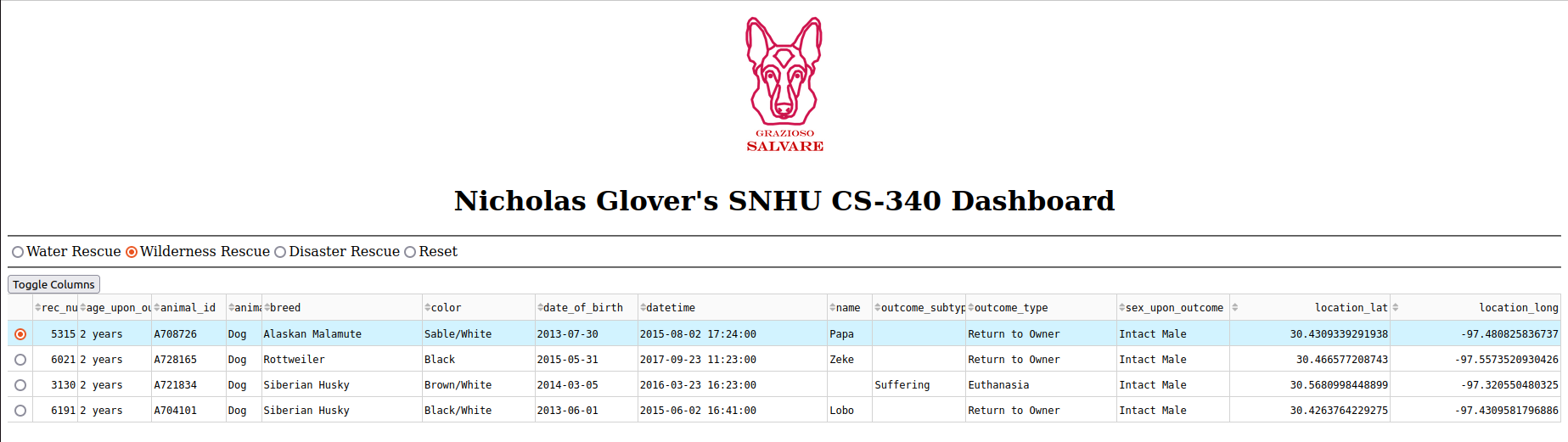


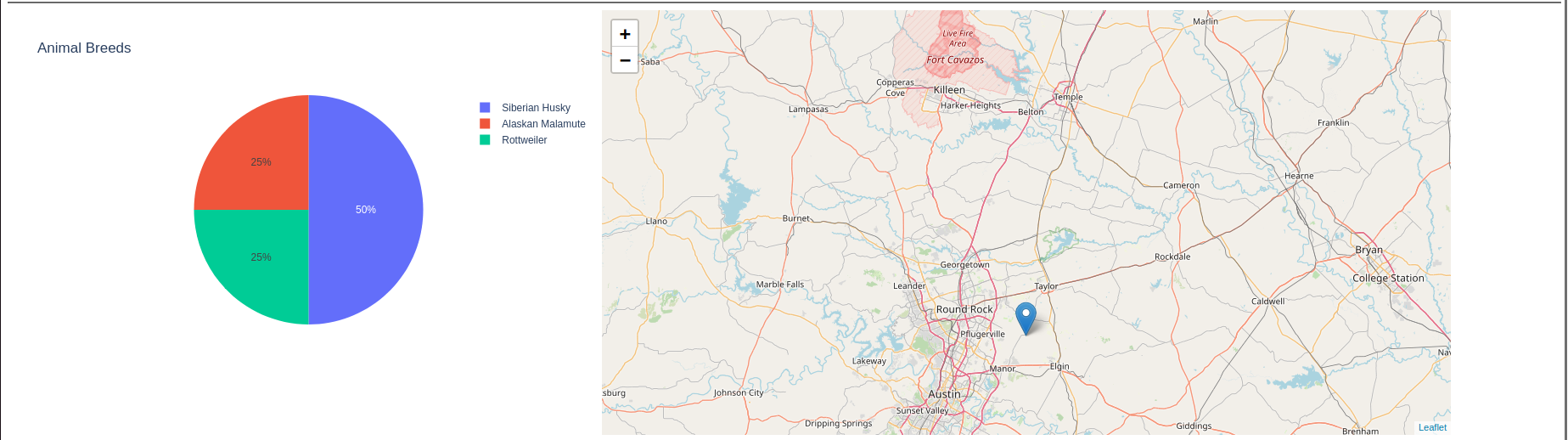
Filter – Water Rescue:



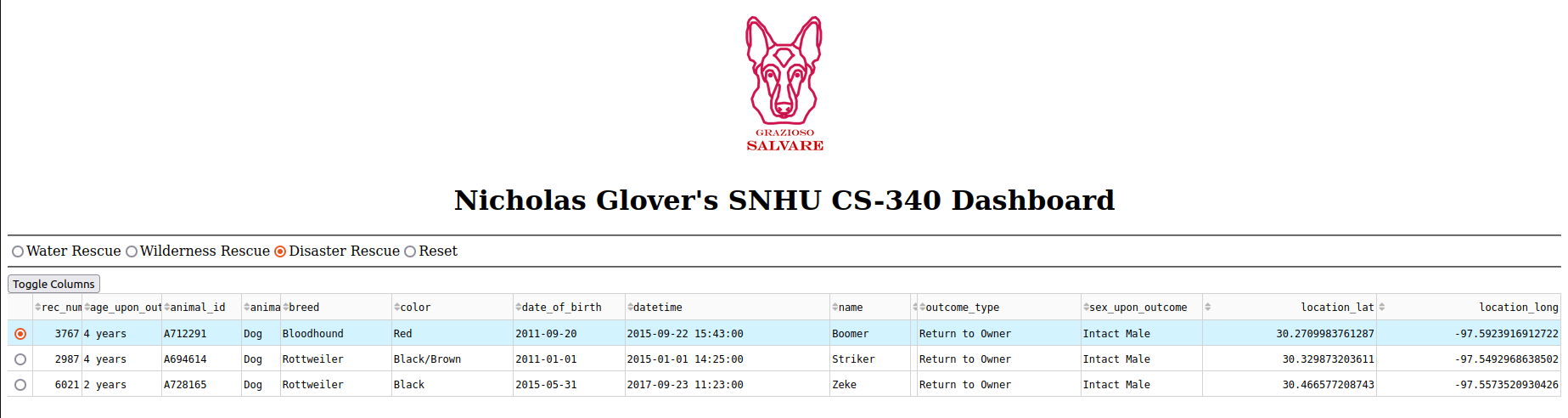


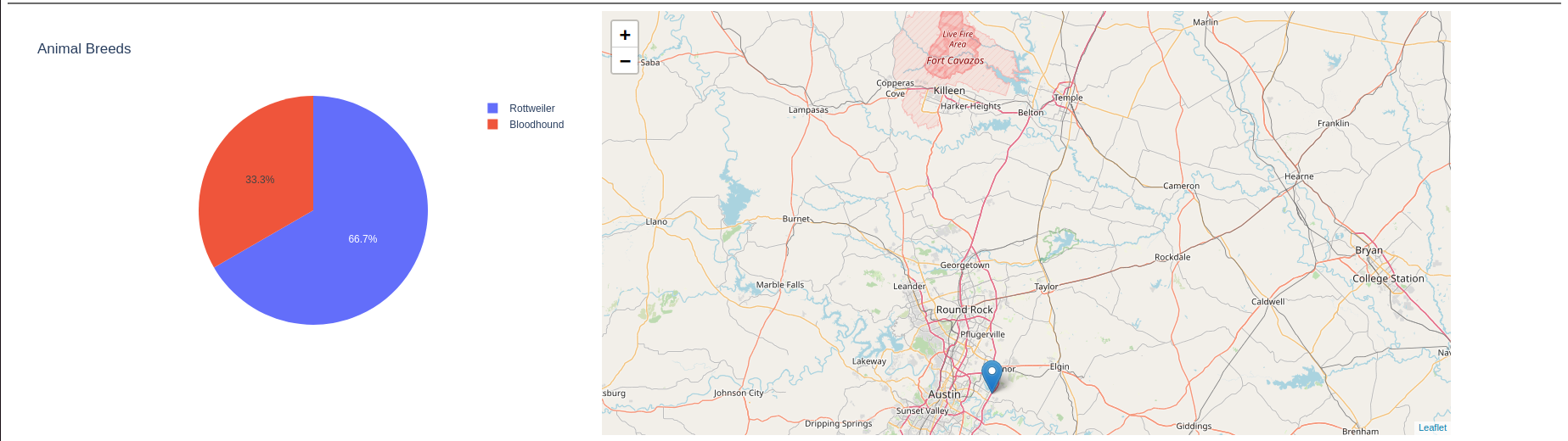
Filter – Wilderness Rescue:





Filter – Disaster Rescue:





## Difficulties

I only had a few difficulties when writing the code for the dashboard. The main issues were when I was trying to set up the pie chart that showed the breeds of the currently represented animals in the database. I wasn’t able to get the correct data to show up, but I fixed that by changing how the data was returned from the callback function.

## Contact

Nicholas Glover

nicholas.glover@snhu.edu