# CS 340 Project 2 README

## About the Project/Project Title

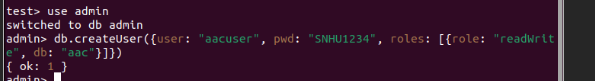
This application allows its users, Grazioso Salvare to effectively search for and filter through their database of animals. The web application displays graphs that may be altered to easily find and display different animals that fit their requirements to be a training animal in various categories. This is paired with a CRUD module that allows permanent modification of the database on the backend.

## Motivation

The client needs to be able to filter through a large database and quickly find animals that suit their training requirements. The users need to be able to filter through the data intuitively.

## Getting Started

* Import your csv file into mongoDB using mongoimport tool following guidelines for your database <https://www.mongodb.com/docs/database-tools/mongoimport/>
* Setup roles and authentication, and any indexing necessary
  + Users can create roles by using the admin database and using “db.createUser({user : ‘<username>’, pwd : ‘<password>’, roles : [{role: ‘<user\_role>’, db: ‘<database’>}]}) and replacing information in angle brackets with your needs.



The user can then authorize the created user by using the command “db.grantRolesToUser(‘<user>’, [{‘role’: ‘<permissions>’, ‘db’: ‘<database>’}])



* Import your python file into a python editor or jupyter notebook & run

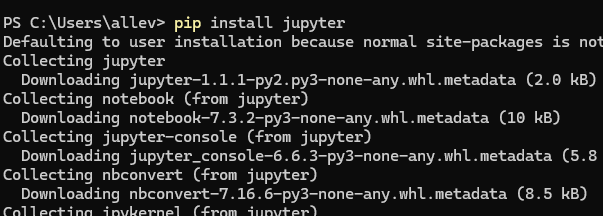
## Installation

Tools Used:

* Python
  + Visit <https://www.python.org/>
  + Download the latest version of Python
  + Run the installer
  + Ensure installation by checking version in terminal/command prompt by entering “python -V”



* Jupyter Notebook
  + Ensure Python is installed
  + Start Terminal/Command Prompt
  + Use Python to install by entering “pip install jupyter”



* MongoDB
  + Visit <https://www.mongodb.com/>
  + Navigate to install free version with your operating system setting
  + Follow through the installer steps

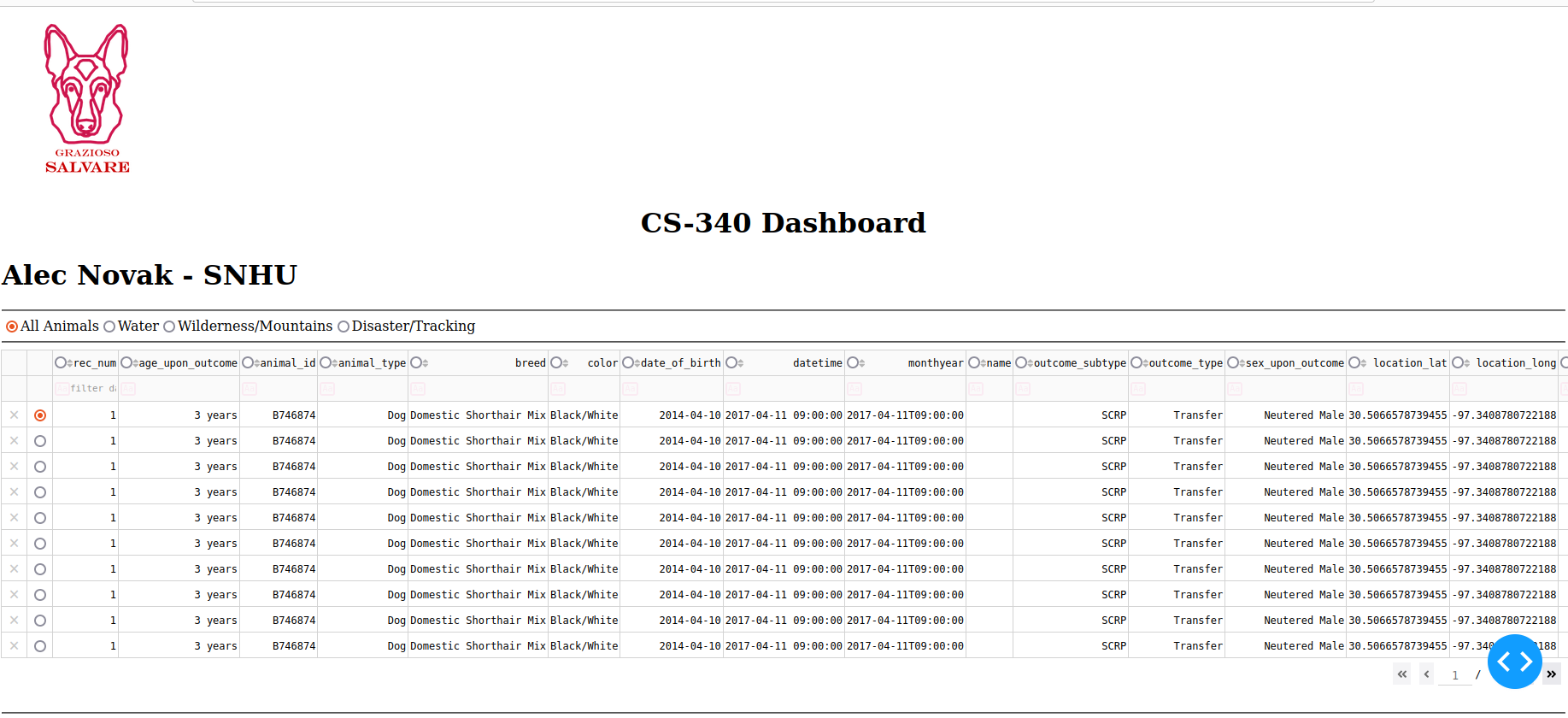
## Usage

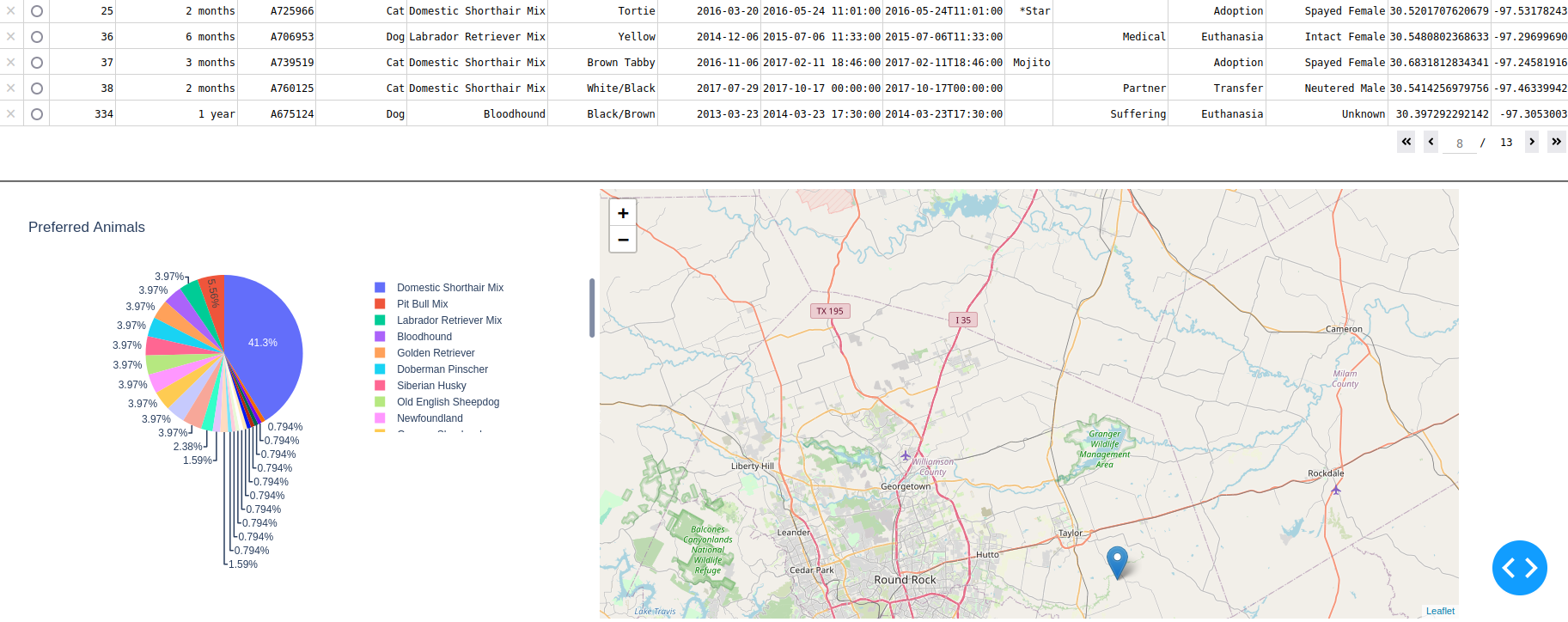
MongoDB was an effective tool in this scenario because it meshes with Python in PyMongo and allows manipulation of a database without using SQL. Combining this with Dash means we can quickly build a web application that allows front-end users to siphon necessary data without having to understand the backend. PyMongo also allows for indexing, which can be useful to reduce the time it takes to return results from common queries. Dash is very useful here, involving leaflet to structure and populate data into more visually appealing graphs that categorize data. Dash allows gives easy access to change the web-app based on inputs using callback functions. In this case, we used callback functions with filtering parameters that filter out animals that don’t match our criteria and feed the same information to update the graphs.

<https://dash.plotly.com/>

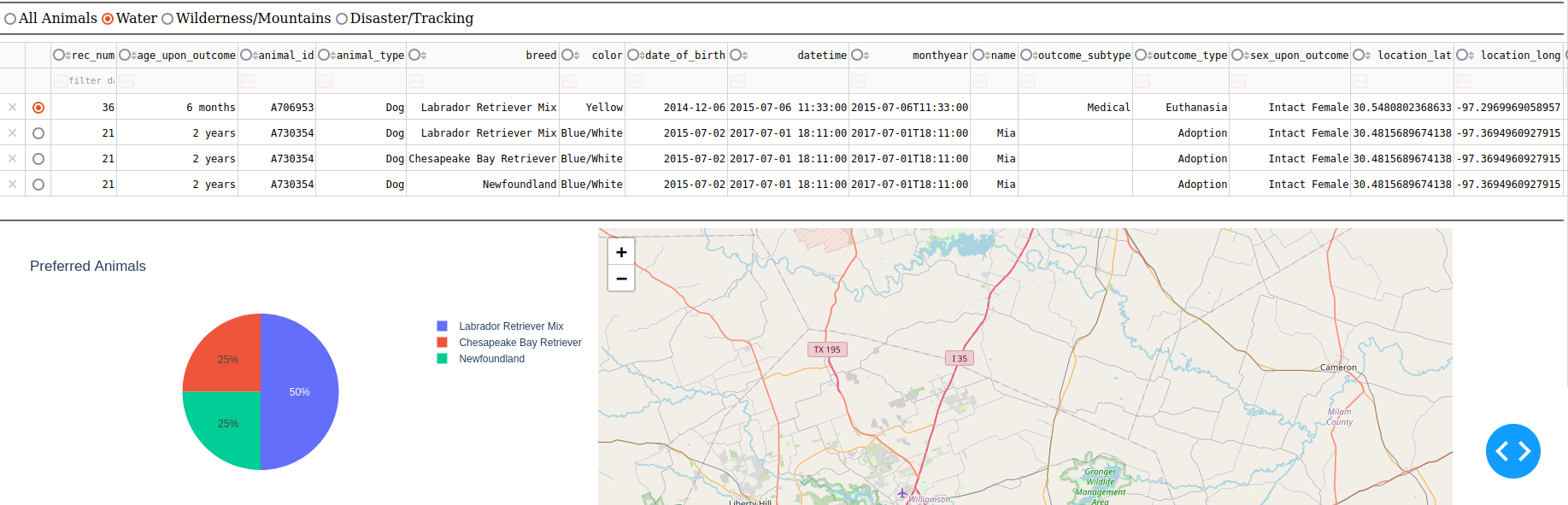
<https://www.dash-leaflet.com/>

**All Animals and initial state:**

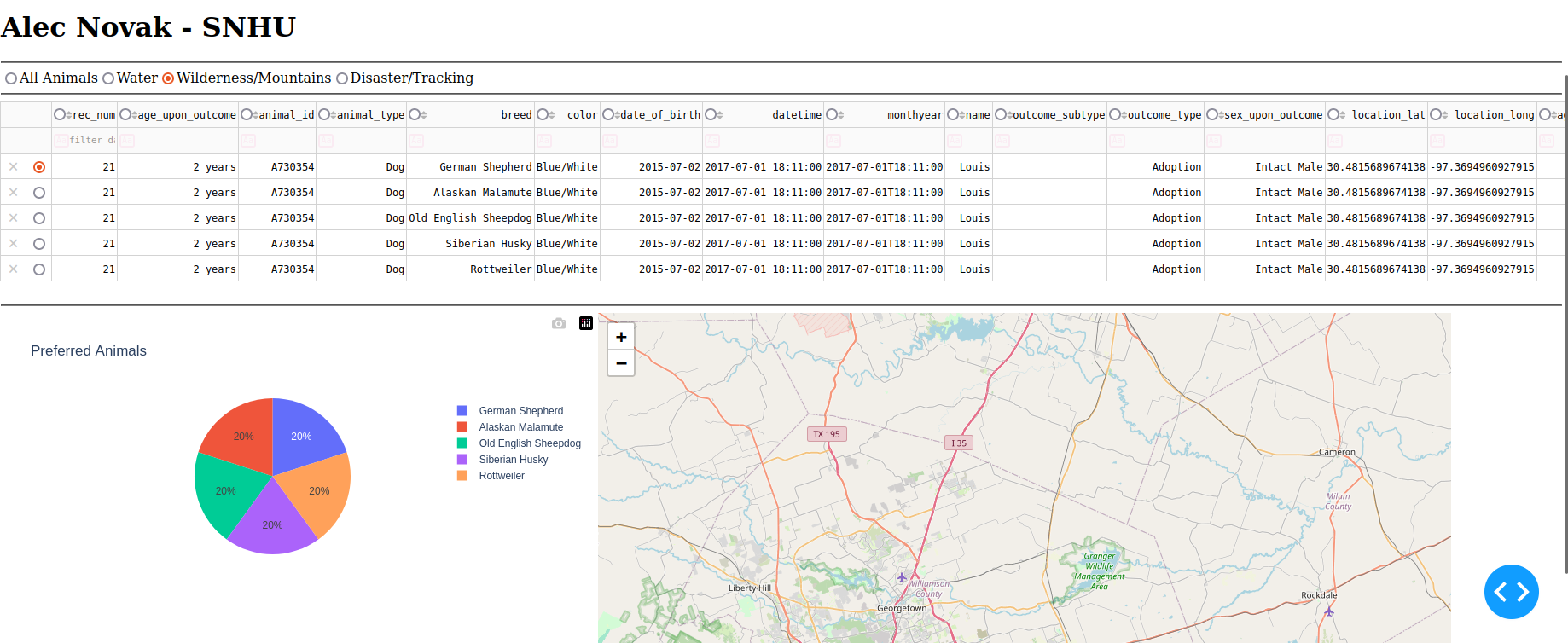




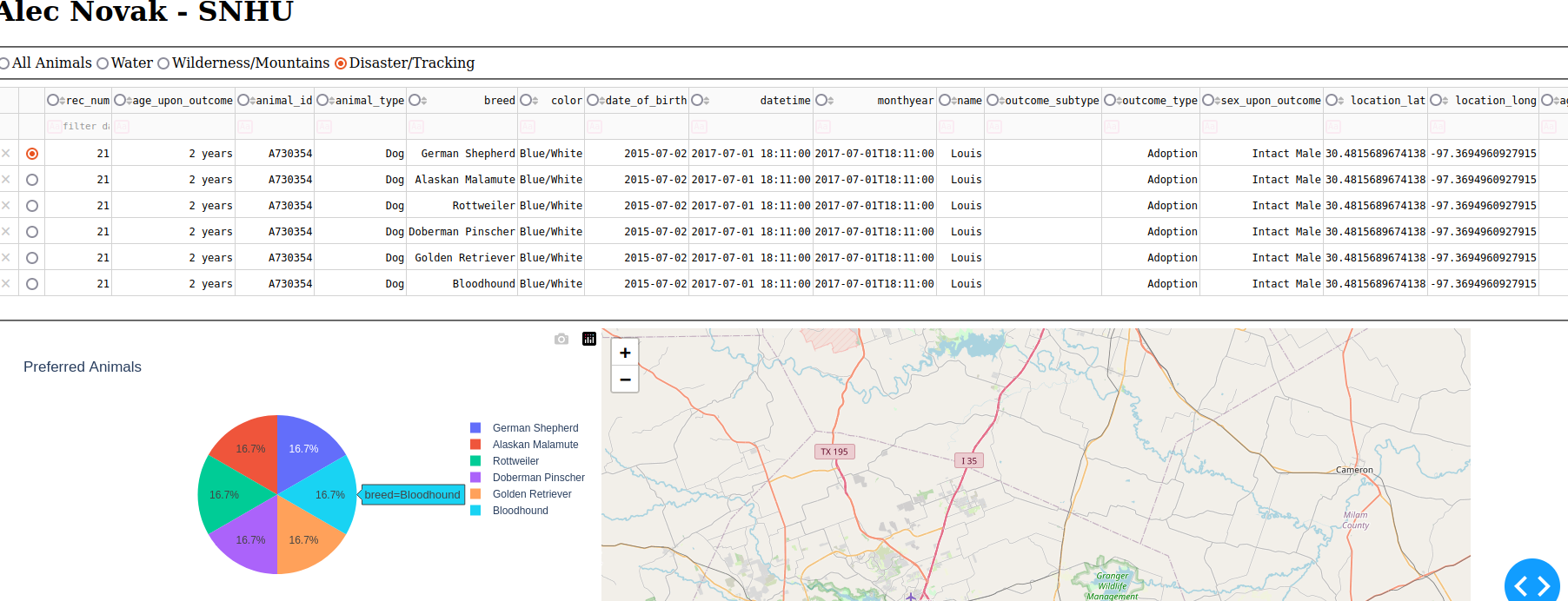
**Water Animals Filter:**



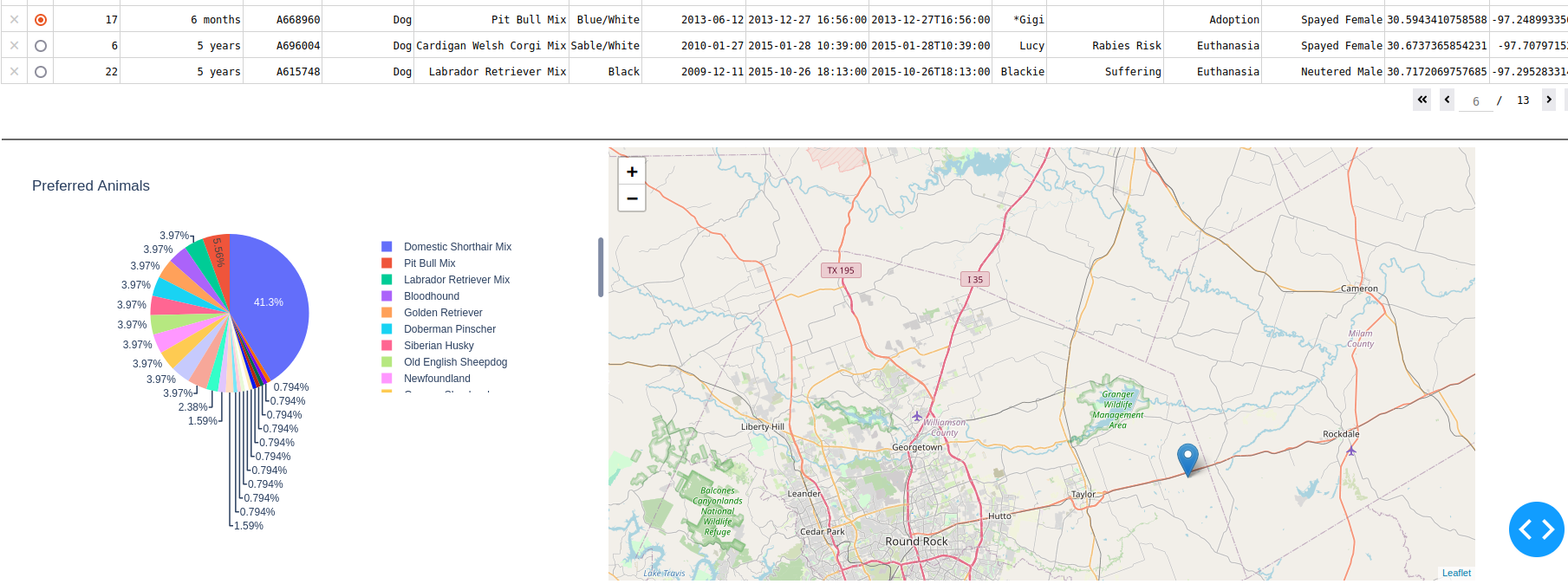
**Wilderness/Mountain Animal Filter:**



**Disaster and Tracking Animal Filter:**



**Animal Geolocation:**



### Code Example

The Project\_One script contains basic CRUD functionality. Users can create an AnimalShelterObject by calling the constructor and passing in the database user and password AnimalShelter(username, password). Then the AnimalShelterObject can be called to use the desired function passing in an animal object with designated attributes. (dict-style data). The create function will insert a new animal with available parameters and return True if successful, or False if the function failed. The read function will return animals with matching criteria that is searched for, which will be an empty list if no objects matching the query are found. The update function can be used by passing in the search parameters as the first argument and the update terms as the second argument. This function will return the number of documents that it affected. The delete method can be used by passing in an identifying query and will delete the documents from the database and will return the number of documents that were deleted.

**Steps and Challenges:**

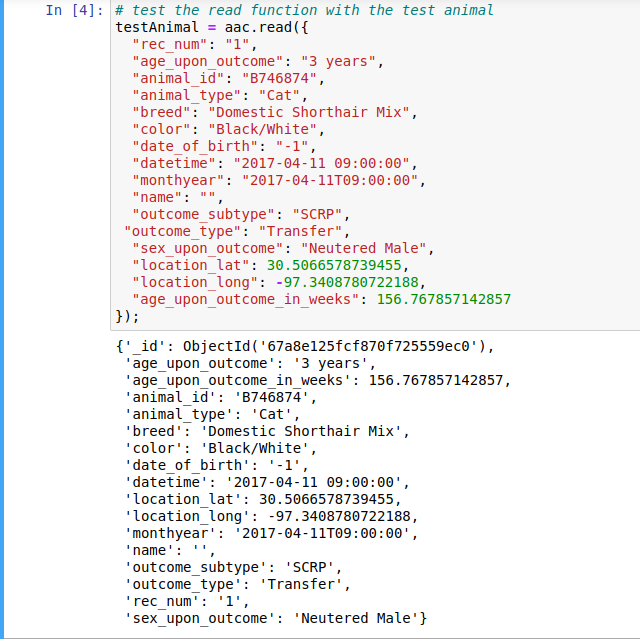
The steps to accomplish the requirements of this project include importing the given CSV file into MongoDB, creating user authentication and providing authorization, connecting the database to a Python application and setting up basic CRUD functionality, and setting up the web-application using Dash. Various tests are run along the way to ensure everything is working as intended. Many challenges appeared during this process. There were some difficulties with importing the CSV file and authorizing the ‘aacuser’ role. More difficulties arose around setting up the server, where the server would try to rerun on the same port, which wasn’t terminated. Shutting down the server and waiting fixed this issue when it arose. The last major difficulties I had were with the database and callbacks. I had accidentally deleted/updated all the database documents when developing this CRUD module, which I attempted to reimport the database (I assume they are actually disconnected because it worked in Mongosh, but not the Jupyter application). This led to an error attempting to drop the “\_id” column, which wasn’t present. I had fixed this by manually copying documents from the database. Lastly, I had some difficulty with the callbacks but eventually chained the pie graph to the filtering callback so that they update together.

### Tests

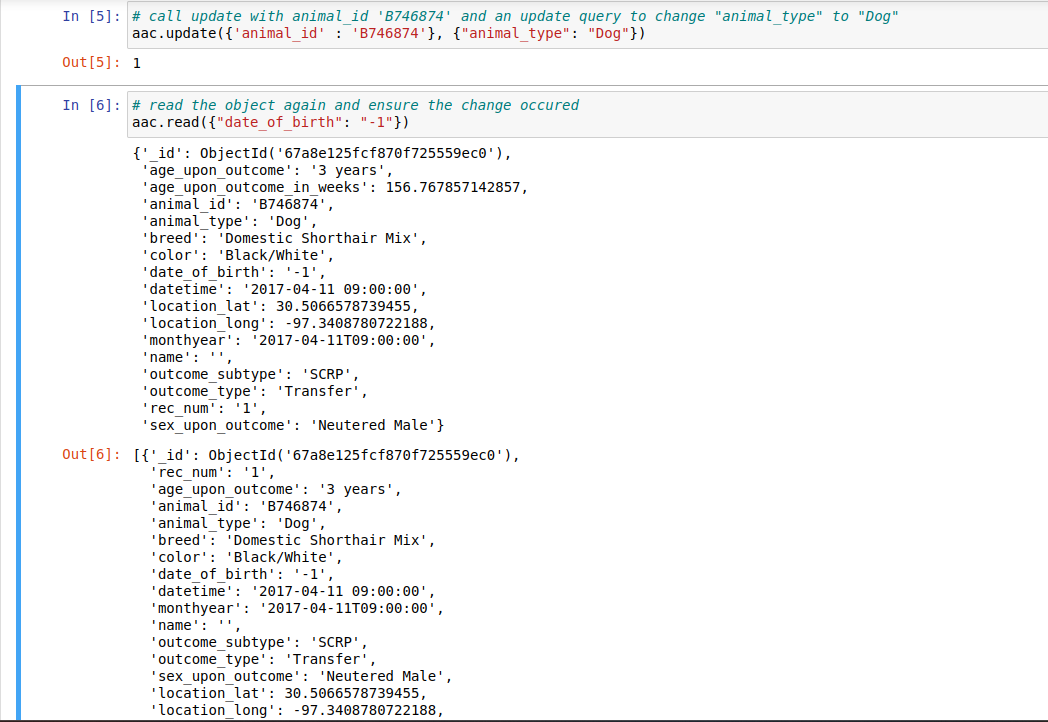
**Create:**



**Read:**



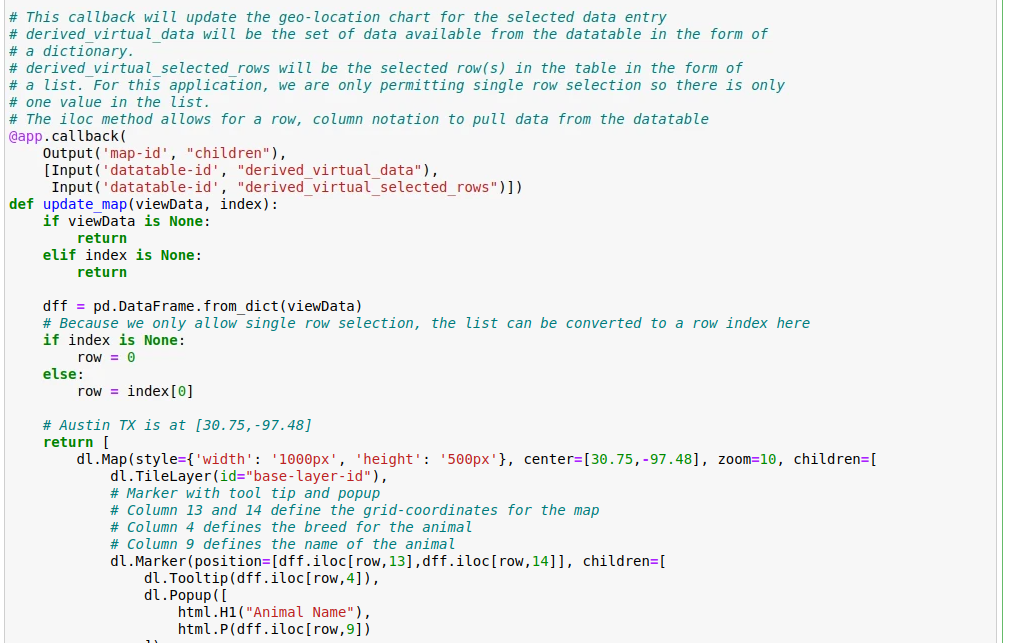
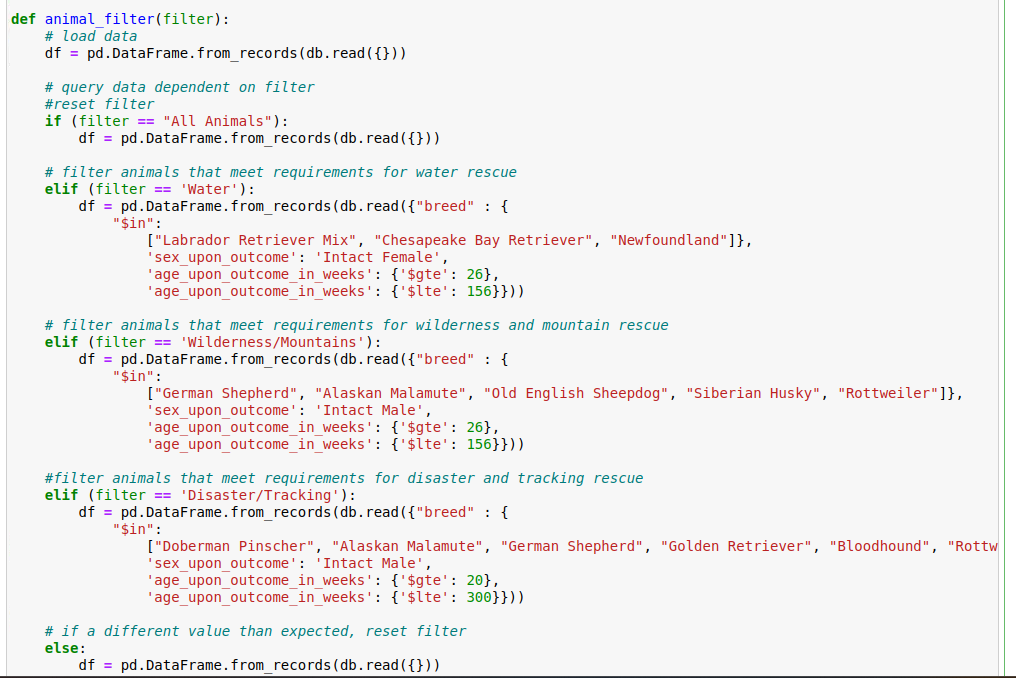
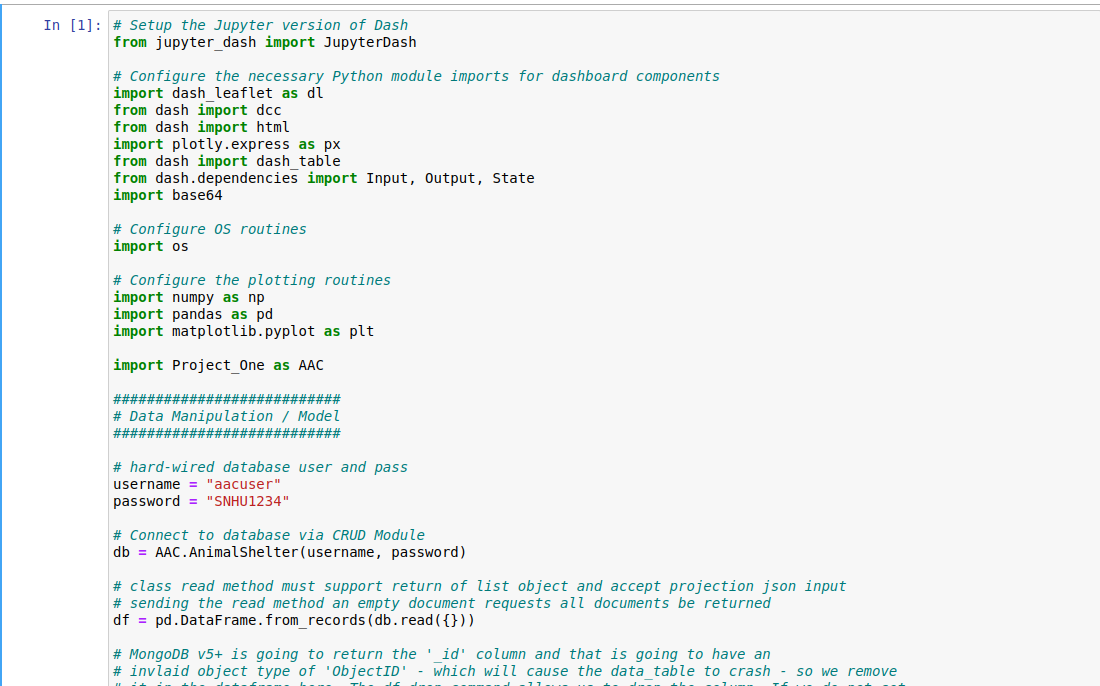
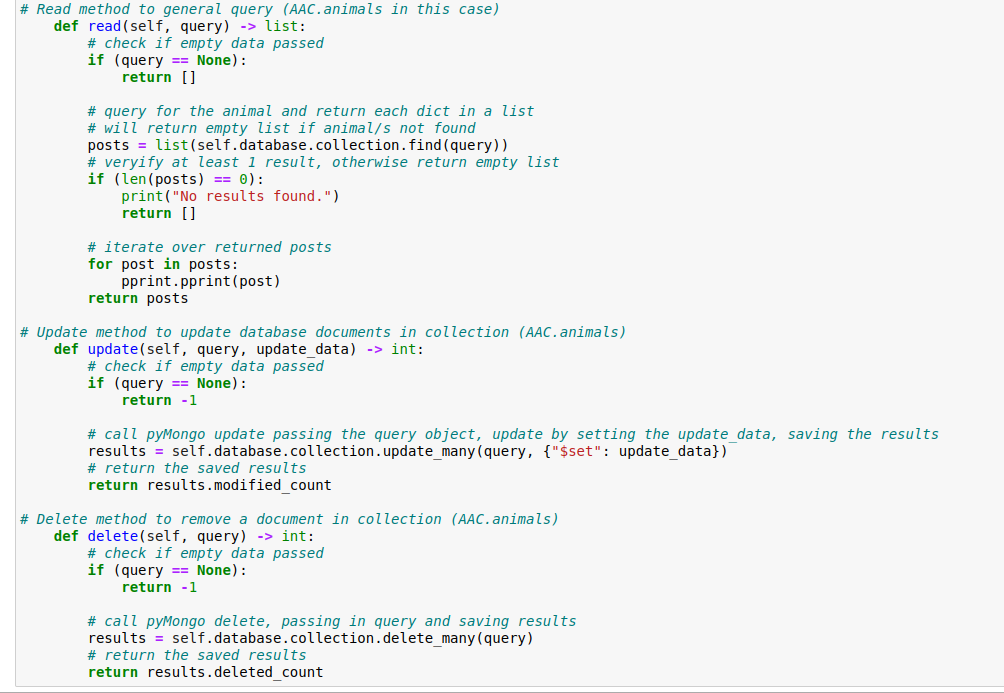
**Update:**



**Delete:**

### 

### Screenshots



## Contact

Alec Novak: